

USER'S MANUAL
INTELLIGENT MOTOR CONTROLLERS
For PCI bus
PCIx FAMILY

OREGON MICRO SYSTEMS, INC.

TWIN OAKS BUSINESS CENTER
1800 NW 169th PLACE, SUITE C100
BEAVERTON, OR 97006
PHONE 503-629-8081
FAX 503-629-0688
EMAIL sales@OMSmotion.com
WEB SITE <http://www.OMSmotion.com>

COPYRIGHT NOTICE

© 2001 Oregon Micro Systems, Inc., A Pro-Dex Company

ALL RIGHTS RESERVED

This document is copyrighted by Oregon Micro Systems, Inc. You may not reproduce, transmit, transcribe, store in a retrieval system, or translate into any language in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, any part of this publication without the express written permission of Oregon Micro Systems, Inc.

TRADEMARKS

IBM, IBM PC, IBM PC/XT, IBM PC/AT, IBM PS/2 and IBM PC DOS are registered trademarks of International Business Machines Corporation. CompactPCI, PICMG-PCI, PICMG are registered trademarks of the PICMG-PCI Industrial Computer Manufacturers Group, Inc. Windows, Win 95, Win 98 and Win NT are registered trademarks of Microsoft Corporation.

DISCLAIMER

Oregon Micro Systems, Inc. makes no representations or warranties regarding the contents of this document. We reserve the right to revise this document, or make changes to the specifications of the product described within it at any time without notice and without obligation to notify any person of such revision or change.

3301-0800000
Revision B

TABLE OF CONTENTS

1. GENERAL DESCRIPTION

INTRODUCTION	1-1
SYSTEM OVERVIEW	1-1

2. GETTING STARTED

PREPARE FOR INSTALLATION	2-1
TO PREPARE FOR INSTALLATION	2-1
INSTALL THE PCIx INTO A PCI SYSTEM	2-5
SOFTWARE INSTALLATION	2-5
FOR WINDOWS NT	2-5
FOR WINDOWS 95/98	2-6
CONNECT TO STEPPER MOTOR SYSTEM	2-6
CONNECT AND CHECKOUT THE SERVO SYSTEM	2-7
CONNECT AND CONFIGURE THE MOTOR/AMPLIFIER	2-8
TUNE THE SYSTEM	2-9
SETTING THE USER DEFAULT CONFIGURATION	2-16
POWER SUPPLY REQUIREMENTS	2-18

3. COMMUNICATION INTERFACE

INTRODUCTION	3-1
PCI INTERFACE	3-1
PCI COMMUNICATION THEORY	3-3
PCI ADDRESS SELECTION	3-4
USING INTERRUPTS	3-4
DATA REGISTER	3-4
DONE FLAG REGISTER	3-4
INTERRUPT CONTROL REGISTER	3-5
STATUS REGISTER	3-6

4. CONTROL SIGNAL INTERFACE

INTRODUCTION	4-1
GENERAL PURPOSE I/O, LIMIT AND HOME INPUTS	4-1
CONTROL OUTPUT	4-1
ENCODER FEEDBACK	4-3
ENCODER SELECTION AND COMPATIBILITY	4-3
HOME PROCEDURES	4-3
IO68-M ADAPTER MODULE	4-6
EXPLANATION OF ADDITIONAL CIRCUITRY ON THE IO68-M	4-7

5. COMMAND STRUCTURE

INTRODUCTION.....	5-1
COMMAND QUEUES.....	5-2
COMMAND SUMMARY.....	5-3
SYSTEM STATUS AND CONTROL COMMANDS.....	5-9
IDENTIFICATION COMMANDS	5-9
POWER-UP DEFAULTS	5-10
COMMAND ECHO.....	5-14
QUEUE SELECTION COMMANDS.....	5-16
QUEUE STATUS COMMANDS	5-22
USER UNIT COMMANDS	5-24
AXIS STATUS COMMANDS	5-27
MACROS	5-32
I/O CONTROL COMMANDS	5-36
AUXILIARY CONTROL COMMANDS.....	5-36
GENERAL PURPOSE I/O CONTROL	5-46
HOME CONTROL COMMANDS.....	5-50
LIMIT CONTROL	5-54
SERVO CONTROL COMMANDS	5-64
SERVO VOLTAGE CONTROL COMMANDS	5-64
PID COMMANDS.....	5-77
STEP ENCODER CONTROL COMMANDS.....	5-90
STEP ENCODER CONTROL COMMANDS	5-90
STEP ENCODER SLIP COMMANDS	5-99
ENCODER SLAVE MODES.....	5-105
HOMING COMMANDS	5-106
POSITION COUNTERS.....	5-111
PROFILE CONTROL COMMANDS.....	5-119
VELOCITY COMMANDS	5-119
ACCELERATION COMMANDS	5-124
PROFILE COMMANDS	5-128
CUSTOM PROFILE COMMANDS	5-136
MOTION GENERATION COMMANDS.....	5-142
JOGGING COMMANDS	5-142
MOVE SPECIFICATION COMMANDS	5-147
MOVE EXECUTION COMMANDS.....	5-151
MOVE TERMINATION COMMANDS.....	5-155
MOVE COMPLETION NOTIFICATION COMMANDS.....	5-162
VELOCITY STAIRCASING COMMANDS	5-168
VELOCITY STREAMING COMMANDS	5-174
CONTOURING COMMANDS	5-175
SYNCHRONIZATION COMMANDS	5-186

WAITING COMMANDS.....	5-186
LOOPING COMMANDS	5-191
LOOPING COMMANDS.....	5-191

6. HOST SOFTWARE

INTRODUCTION TO PCIX SOFTWARE SUPPORT	6-1
---	-----

7. SERVICE

USER SERVICE	7-1
THEORY OF OPERATION	7-1

A. LIMITED WARRANTY

B. TECHNICAL INFORMATION / RETURN FOR REPAIR PROCEDURES

C. SPECIFICATIONS

INDEX

This page intentionally left blank

1. GENERAL DESCRIPTION

1.1. INTRODUCTION

The Oregon Micro Systems PCIx family of PCI products comply with the PCI Local Bus specification (revision 2.2) for a "short-card". The PCIx controller can manage up to four axes of stepper or servo motion control while incorporating other signals; i.e. limits, sensors, I/O, etc. into the system. It can manage coordinated or independent motion of each or all of the axes simultaneously. With high level functionality, such as circular and linear interpolation, multi-tasking, custom profiling, etc., the PCIx can satisfy most any motion control application. [See Appendix C Specifications](#) for specific PCIx Family Models.

The PCIx communicates as a slave only device and functions as a motion coprocessor. It utilizes a 32-bit microprocessor and patented, proprietary technology to control the trajectory profile, acceleration, velocity, deceleration and direction of selected axes. In response to commands from the host computer, the PCIx controller will calculate the optimum velocity profile to reach the desired destination in the minimum time while conforming to the programmed acceleration and velocity parameters. In addition the PCIx can provide motion control information such as axis position, the state of overtravel limits and Done interrupts.

The stepper control of the PCIx produces a 50% duty cycle square wave step pulse at velocities of 0 to 1,044,000 pulses per second and an acceleration of 0 to 8,000,000 pulses per second, per second. The servo control utilizes a 16-bit DAC and outputs either +/- 10V or 0 to +10V. The encoder feedback control can be used as feedback for the servo PID, position maintenance for the stepper axes or as strictly a position feedback of any axes. The encoder input supports either differential or single ended quadrature TTL signals at a rate of up to 4MHz and counts at a 4 times resolution. This means a 1000 line encoder will produce 4000 counts per revolution in the PCIx controller.

The PCIx command set employs an ASCII character syntax. Using virtually any programming language, simple ASCII command strings are sent to the PCIx through the PCI bus. A typical motion requirement of 1,000,000 counts at 400,000 counts/sec and an acceleration of 500,000 counts/sec/sec the following string would be sent from the host computer to the PCIx:

VL400000; AC500000; MR1000000; GO

For additional command programming examples see Section 5 Command Structure

1.2. SYSTEM OVERVIEW

The PCIx is physically a standard PCI module, with plug and play capability. The PCIx communication interface is accessed through the bus.

The PC1x utilizes the Motorola 68332 32-bit microcontroller and FPGA technology for extensive logic integration and flexibility. The firmware, which resides in Flash Memory, can be upgraded through the communication interface without having to remove the controller from the system. All input signals to the PC1x are buffered through differential or opto-coupled components and are located on connector J2. The PC1x utilizes the +5VDC supplied by the Host computer through the PCI bus for the logic control and the servo models utilize the additional +/- 12V supply for the analog control.

The PC1x supports four I/O address registers that provide near real-time information. The data communication is performed by sending and receiving strings of data (ASCII characters) through the data port register. The status register provides handshaking information for writing to the data register as well as some status information including error conditions, motion complete, etc. The PC1x can generate an interrupt to the PCI host and the conditions that cause interrupt can be individually selected by writing to the control register.

More details on each of the communication interfaces as well as the functionality of the controller are included in the following chapters.

2.

GETTING STARTED

2.1. PREPARE FOR INSTALLATION

The installation of the PCIx board is straight forward: With power OFF to the computer, insert the card in a PCI slot of the computer. Power up the computer, and install the device driver.

NOTE: The PCIx board may not work well if it is assigned to a shared IRQ in the host system. If this is the case, disable the shared IRQ in the host system's BIOS before installing the PCIx.

Though the PCIx is a very low power device, there should be ventilation, including forced air, around the circuit board. The PCIx will get all of its power from the PCI bus.

2.2. TO PREPARE FOR INSTALLATION

If you plan on changing any of the factory default jumper or switch settings this should be done before installing the PCIx in the computer. The I/O Bit Direction switch and Board Number Select jumper must be set BEFORE power up.

If the PCIx is to be installed with other PCIx boards, the Board Number Select jumper must be set to a different value on each board. The Board Number Select jumper is located near the top center of the PCIx board and is labeled JP6 ([see Figure 2-5 on page 2-4](#)). Set the first PCIx to identification number 1, the second PCIx to 2, and so forth, according to [Table 2-1 Possible Board Number Select Settings \(JP6\)](#). [See Section 2.3 Install the PCIx into a PCI System](#) for further installation instructions. For instruction on configuring stepper motors for use with the PCIx [see section 2.5 Connect to Stepper Motor System](#). For instruction on configuring servo motors with the PCIx, [see section 2.6 Connect and Checkout the Servo System](#).

Switches S1 and S2 are used for encoder biasing when using single ended encoders. For differential encoders, leave these switches in the OFF position (factory default). When using single ended encoders on an axis, put the appropriate switch positions for Phase A-, Phase B- and Index- of that axis in the ON position. See [figure Figure 2-1 Definition of Encoder Biasing Switches](#).

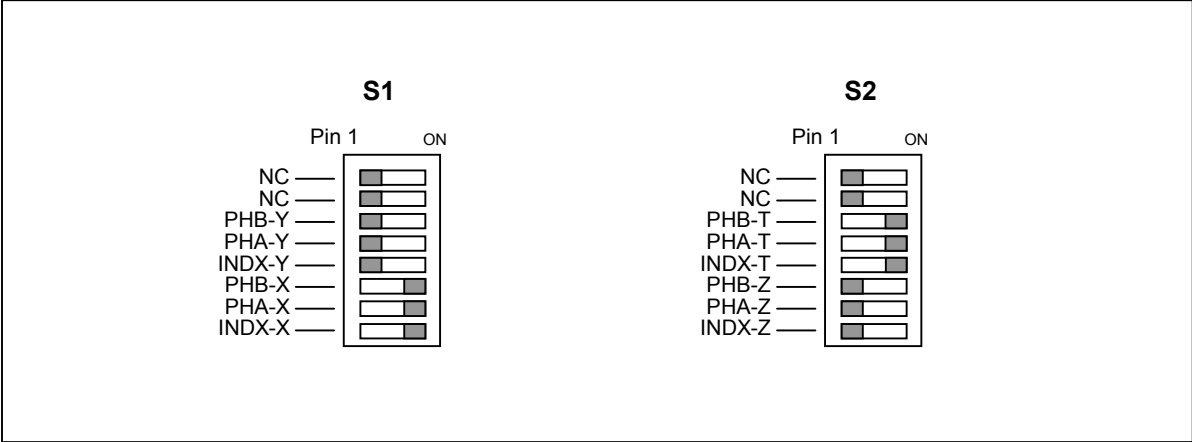


FIGURE 2-1 DEFINITION OF ENCODER BIASING SWITCHES

Note: In [Figure 2-1](#) switches S1 and S2 are configured for differential encoder inputs for the Y and Z axes and single ended encoder inputs for the X and T axes.

Switch S3 is used to determine I/O bit direction. The factory default setting (shown in [Figure 2-5 PCIx Jumpers/Switches](#)) sets I/O bits 0-3 as inputs and bits 4-7 as outputs. A switch in the ON position sets the associated I/O bit as an output. A switch in the OFF position sets the associated I/O bit as an input. See [Figure 2-2 S3 Switch Definition](#).

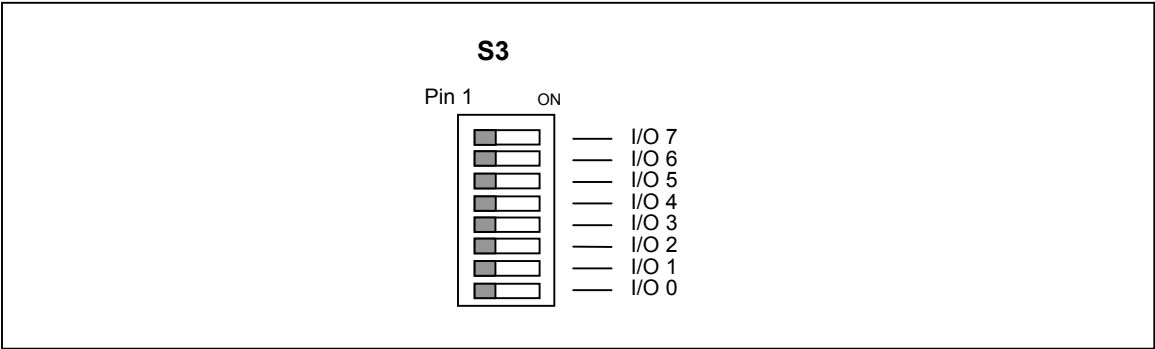


FIGURE 2-2 S3 SWITCH DEFINITION

Note: In [Figure 2-2](#) switch S3 is configured for I/O bits 0-7 as inputs.

Jumper J3 is used to route the appropriate I/O bit to the output connector J2. The factory default setting routes the input bits 0-3 and the output bits 4-7 to J2. See [Figure 2-3 J3 Pin Definition](#).

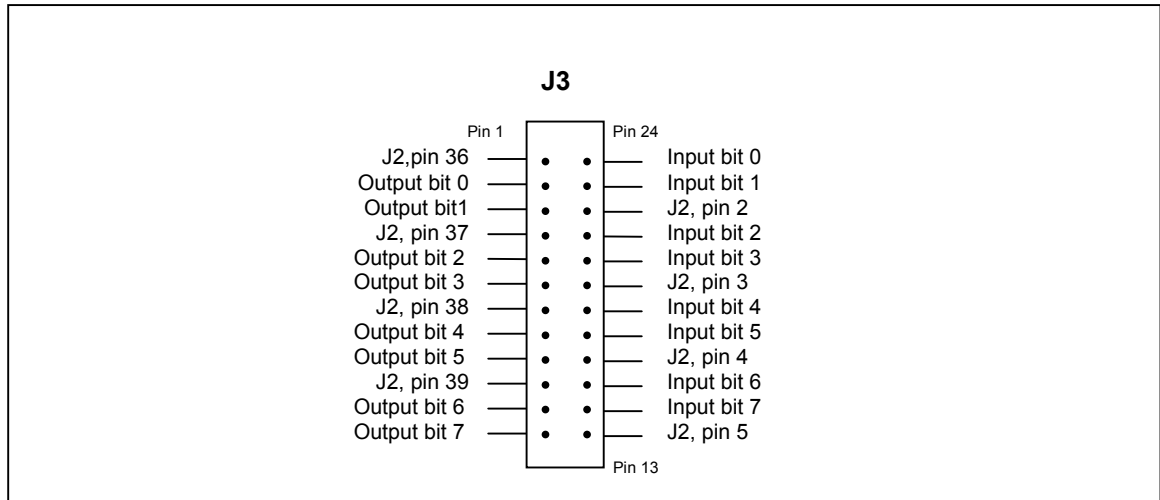


FIGURE 2-3 J3 PIN DEFINITION

For example, if you set all I/O bits as inputs, J3 would be configured as follows:

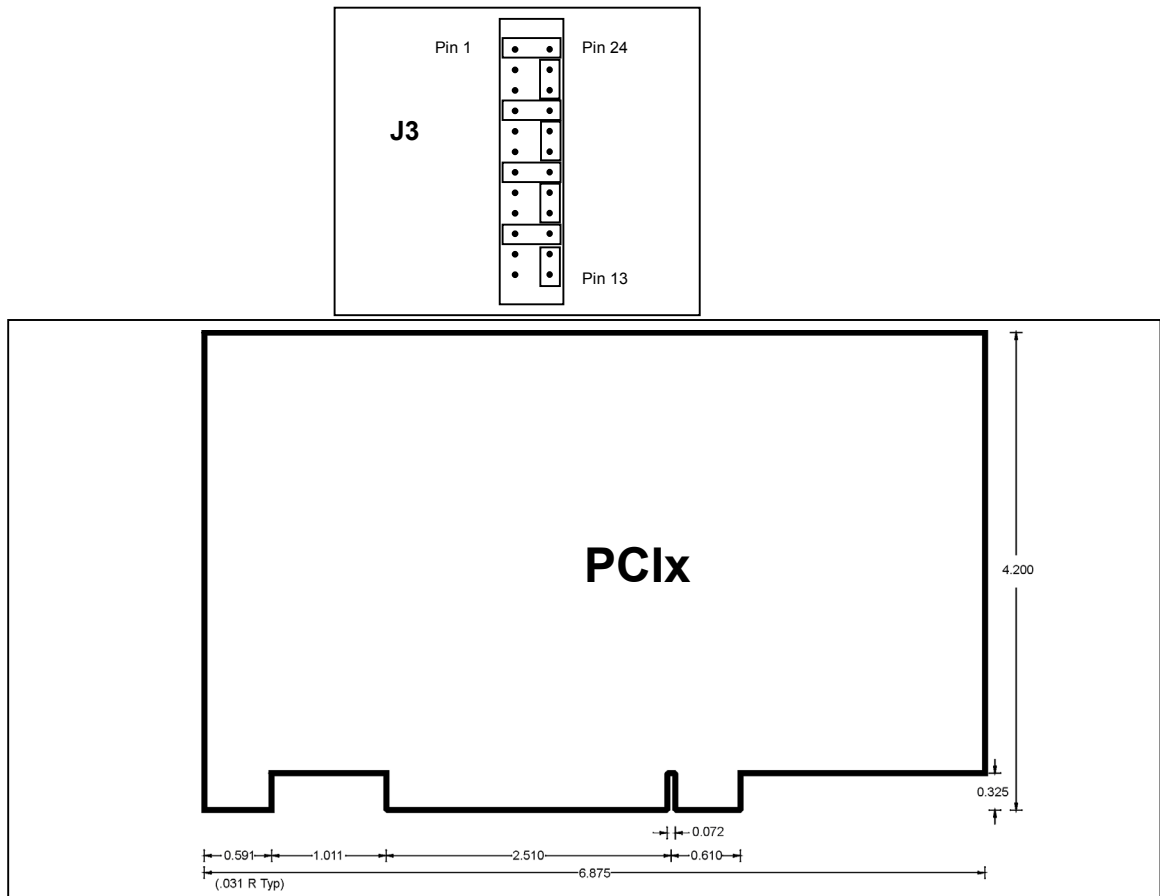


FIGURE 2-4 PC1x DIAGRAM

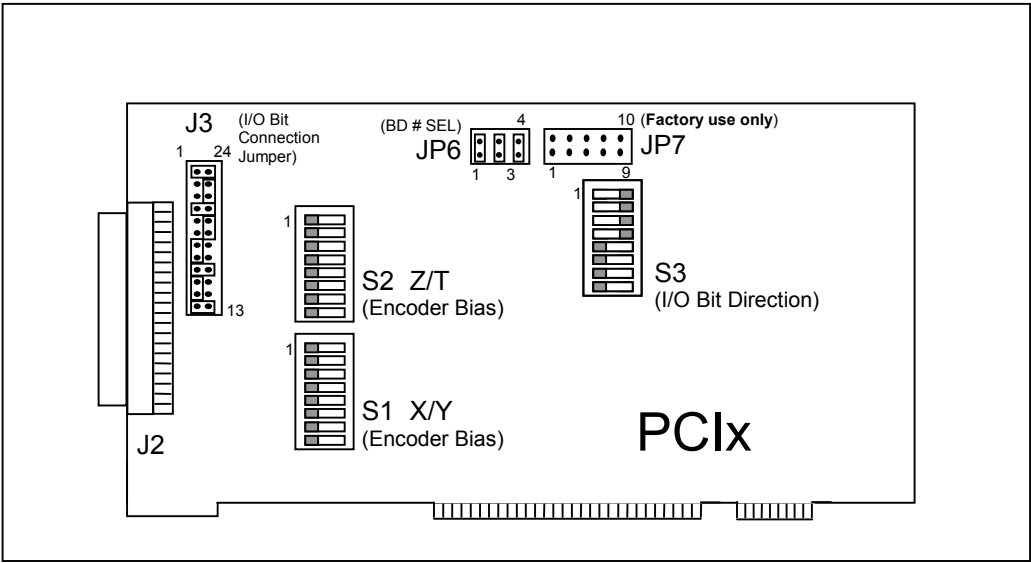
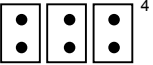
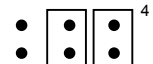


FIGURE 2-5 PC1x JUMPERS/SWITCHES (Factory default shown)

Figure 2-4 illustrates the locations of the jumpers and switches on the PC1x.

TABLE 2-1 POSSIBLE BOARD NUMBER SELECT SETTINGS (JP6)

Board ID#	Jumper 1 / 2 / 3 Setting
1*	On / On / On 
2	Off / On / On 
3	On / Off / On 
4	Off / Off / On 

* Factory Default

Up to four PC1x boards may be installed in the same computer. Note that Jumper 3 in the Board Number Select (JP6) must be ON at all times.

2.3. INSTALL THE PCIX INTO A PCI SYSTEM

- Turn off power to the PCI computer, and disconnect its power cord from the wall socket. Remove the cover of the computer to gain access to the PCI Bus.
- On the PCIX Board, use factory default jumper settings or configure all switches and jumpers as needed. (Configurable switches are: S1, S2 and S3. Configurable jumpers are: J3 and JP6).
- Align the PCIX in the PCI slot of the computer and insert the PCIX fully into the slot (non-master slot.)
- Double check the board to ensure it is properly seated in the connector.
- Use a screw to fix the bracket on the PCIX to the computer's chassis.
- Replace the power cord, and turn on the computer.
- Allow the computer to boot up.

Caution

Establish communication with the controller board **before** wiring external components to the board (i.e. drivers and motors).

DO NOT make wiring connections to the controller board with power applied to the board.

2.4. SOFTWARE INSTALLATION

OMS provides drivers for Windows NT and Windows 95/98, for other operating systems please contact Oregon Micro Systems, Inc., [refer to Appendix B on page B-1](#).

2.4.1. FOR WINDOWS NT

After installing the PCIX in the chassis, apply power to the host PC and insert the software support disk or CD-ROM supplied by OMS or downloaded from the OMS Website. Follow the installation instructions found in README.TXT in the root directory of the disk.

To begin communicating with the PCIX, run PCICOMM.EXE from the installation disk. You can begin interactively sending commands and receiving responses immediately if all has been properly installed. If the Board Number select jumper is set to something other than 1, run the communication program with the switch /c:#, where # is the Board Number. For example if Board Number = 2 then type "PCICOMM /C:2".

Type WY and observe the response from the PCIx. You should receive a reply similar to "PCIX-400 ver 1.00 S/N 000000" from the PCIx. If you receive nothing, double check that the PCIx is fully seated in the chassis and the device drivers are installed properly. For technical support, [refer to Appendix B on page B-1](#) of this manual for contact information.

2.4.2. FOR WINDOWS 95/98

After Installing the PCIx in the chassis, apply power to the host PC. The system will request a driver for the device. Insert the software support disk from OMS and follow the instructions in Windows for adding a new hardware device.

To begin communicating with the PCIx, run PCICOMM.EXE from the installation disk. You can begin interactively sending commands and receiving responses immediately if all has been properly installed. If the Board Number select jumper is set to something other than 1, run the communication program with the switch /c:#, where # is the Board Number. For example if Board Number = 2 then type "PCICOMM /C:2".

Type WY and observe the response from the PCIx. You should receive a reply similar to "PCIX-400 ver 1.00 S/N 000000" from the PCIx. If you receive nothing, double check that the PCIx is fully seated in the chassis and the device drivers are installed properly. For technical support, [refer to Appendix B on page B-1](#) of this manual for contact information.

2.5. CONNECT TO STEPPER MOTOR SYSTEM

The PCIx control signals are located on the J2 connector. This section is going to explain how to connect a stepper motor driver to the controller board.

Begin this procedure with an OMS controller board installed in your system. Be sure that communication to the board has been established. This can be checked by issuing a WY command to the board, and verifying that the board responds with its model type and revision level (i.e. PCIX-400 ver 1.00 S/N 000000).

NOTE: [Reference section 2.4 Software Installation on page 2-5](#) for software installation instructions.

Once communication has been established with the controller, shut down the system and turn power off to the controller board.

Connect the motor phase signals from the motor to the stepper driver output signals. Use the motor and stepper driver manufacturer's manuals for instructions.

Now, connect the controller signals from J2 on the PCIx SCSI connector, to the stepper driver. Short cable lengths and shielded cables are recommended for improved signal integrity and reduction in signal noise.

If you are using the IO68-M, connect the IO68-M to the PCIx using a shielded cable with 68-pin connectors on either end. From the terminal block on the IO68-M connect wires to your motor drivers and system I/O.

Attach the STEP outputs from the controller to the STEP inputs on the stepper driver. Do the same for DIR signals.

Next, connect an external power supply (that is OFF) to the stepper driver. Again, refer to the manufacturer's manual for instructions. (Note that power supply requirements differ from driver to driver.)

Once all wire connections have been made, power can be restored to your system. It is recommended that you bring the controller board up first (so it is in a known state), and then apply power to the stepper driver.

Refer to [Figure 2-6](#) for an example wiring diagram of OMS' PCIx-004 connected to the OMS PMD4-m stepper driver.

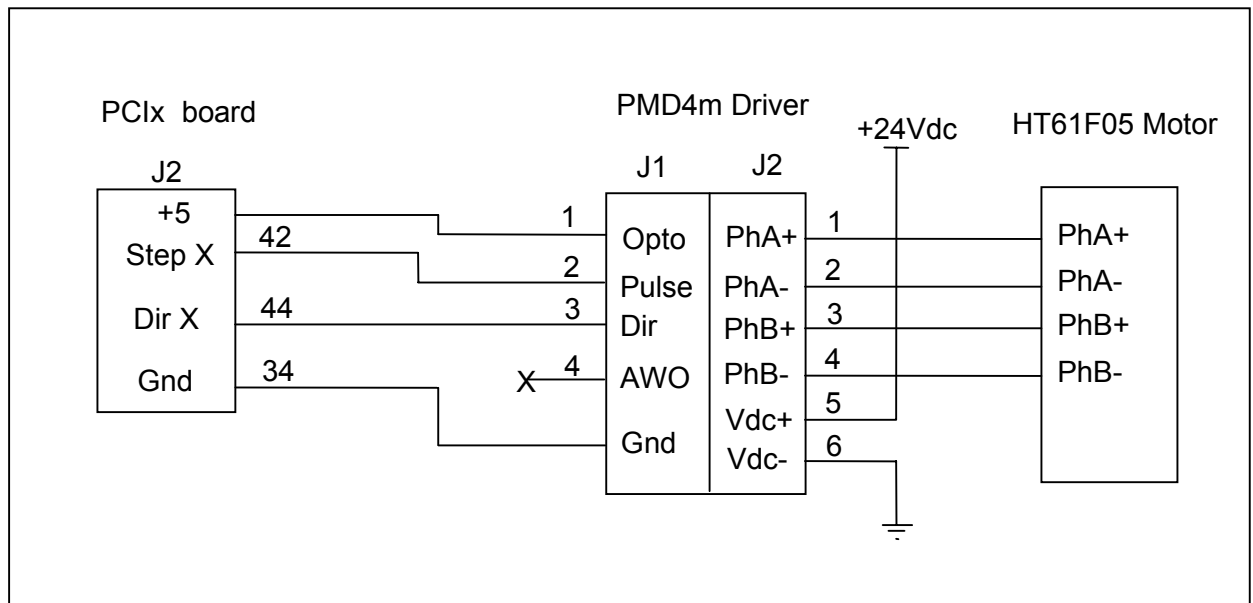


Figure 2-6 Example Wiring Diagram of PCIx Controller

2.6. CONNECT AND CHECKOUT THE SERVO SYSTEM

Servo systems tend not to respond gracefully to connection errors. You can reduce the chance of making connection errors by following a step-by-step procedure:

Caution

The servo motor may jump or spin at a very high velocity during connection and configuration. The motor should be restrained via fastening to the physical system or by some other means before beginning this procedure. Keep hands and clothing clear of the motor and any mechanical assemblies while performing this procedure.

2.6.1. CONNECT AND CONFIGURE THE MOTOR/AMPLIFIER

1. Connect and configure your amplifier per the manufacturer's instructions for "Torque" or "Open-Loop" mode.
2. With the motor and amplifier power turned off, connect the PCIx to the amplifier.
3. Balance your motor:
 - a. Using a voltage meter, verify that the command signal from the PCIx is less than 500mV. If it is not, send the command "DZ0;" to the PCIx and recheck the voltage. If the voltage is still too high, contact Oregon Micro Systems' Technical Support department for guidance.
 - b. Turn on power to the amplifier and then to the motor.
 - c. Adjust the balance setting of your amplifier (if equipped) until the motor stops moving.
 - d. If the motor continues to revolve or your amplifier has no balance adjustment:
 - i) Send the command "DZ100;" to the PCIx.
 - ii) If the motor spins faster, reduce the command parameter and resend the command, e.g. "DZ50;".
 - iii) If the motor spins slower but does not stop, increase the command parameter and resend the command, e.g. "DZ150;".
 - iv) Continue adjusting and resending the DZ command until the motor comes to rest. Write down the final DZ value for later reference as your "zero" setting.
4. Maximize your system's usage of the PCIx's DAC:
 - a. Connect the servo encoder to the PCIx.
 - b. Set the signal/command gain of your amplifier to it's minimum setting.
 - c. Send the "DZ3277;" command to the PCIx and observe the velocity of the motor. The output of PCIx will be near 1VDC.

- d. If the motor does not move at all, your amplifier does not work well at a low velocity. In this case, adjust the signal/command gain of the amplifier to approximately 20% of maximum or until the motor begins to move.
 - e. Using a frequency meter, measure the pulse rate of Phase A of the encoder. The frequency measured is $\frac{1}{4}$ of the actual pulse rate.
 - f. Adjust the signal/command gain of the amplifier until the pulse rate of Phase A (4 is approximately 10% of your desired peak operational velocity. If the pulse rate is already greater than 10% of peak, your amplifier is not designed for low velocity motion and you will likely have some difficulty tuning your motors.
 - g. Send the “DZ-3277;” command to the PCIx and recheck the velocity. You may need to readjust your amplifier. If so, do not reduce the signal/command gain – only increase the setting as needed. Increasing the gain will not impair the forward peak velocity but reduction will.
 - h. Send the DZ command with the “zero” value noted at the end of step 3d (iv) to the PCIx. Send the same value using the KO command, e.g. “KO-175;”.
5. Verify the direction of your servo encoder:
- a. Send the “LP0;DZ2000;” command to the PCIx.
 - b. Send the “RE” command to the PCIx and observe the response.
 - c. If the response is positive, no further action need be taken; go to step 6.
 - d. If the response is negative, your encoder must be reversed.
 - i) If your encoder produces a differential signal, swap Phase B with Phase B-not and repeat from step (a.) above.
 - ii) If your encoder produces a single-ended (or TTL) signal, swap Phase A with Phase B and repeat from step (a.) above.
 - e. If the RE response is still negative, contact OMS Technical Support for assistance.
6. Repeat from step 1 for the other servo axes.
7. Remember to set DZ and KO for each axis at every power-up unless you store the values in flash (see [Section 5 Command Structure](#))

NOTE: Most encoder problems are caused by lack of power or incorrect connections. If the encoder position changes by only 1 count, this is an indication that one of the phases is not connected.

Do not proceed until you perform all the steps in this procedure, ensure that the outputs of the PCIx are as described, and ensure that the encoder is operating correctly

2.6.2. TUNE THE SYSTEM

2.6.2.1. Introduction

The following is an introduction to tuning a servo motor and the basics of the process of doing it. Tuning a servo system is the process of balancing

three primary gain values P, I and D in order to achieve optimum system performance.

In a closed loop system, an error signal is derived, amplified, then supplied to the motor to correct any error. Clearly, if a system is to compensate for infinitely small errors, the gain of the amplifier needs to be infinite. Real world amplifiers do not possess infinite gain; therefore, there is some minimal error which cannot be corrected.

The three primary gain values used in servo systems are P (proportional), I (integral) and D (derivative). The "P" term is used as a straight gain factor to get the system response "in the ballpark." The "I" term defines how quickly the system will respond to change. The "D" term determines the system's stability. This term defines how quickly the system settles at its desired position without oscillating.

The effects of these parameters can be seen when looking at the system's response to a step change at the input. The shape of the step response falls into one of three categories: under damped, critically damped or over damped. Over damped systems are slow to reach their final value and produce little or no oscillation. Critically damped systems reach final value quickly, without overshoot. Under damped systems reach final value quickly, but have various degrees of "ringing", or oscillation, that decay to zero over time. Ideally, a system should be critically damped, allowing for the fastest response time with the least amount of oscillation.

2.6.2.2. Manual Tuning

In most all motion control applications the optimum tuning of the servo system is achieved through a manual tuning process. Auto-tuning algorithms typically can only get the system parameters close and require manual steps to fine tune the parameters. An empirical trial and error approach will be discussed first.

There are some system parameters that need to be defined before attempting to tune a motor. The encoder resolution, counts per revolution, is one element to be determined. Another is the system's maximum velocity. Note that a motor should never exceed 90% of the motor's top rpm. If the system requirement is for a velocity higher than 90% of the motor's top rpm, then another motor with higher rpm capability is to be used.

The system's maximum acceleration is determined several different ways. The best method is to determine the system time constant, which includes "hitting" or "bumping" the motor under system load and measure the time from 0 rpm to maximum rpm and divide this value by 5. The maximum acceleration is either 2.5 times this value, or is based on the system requirements for handling the load as defined in the operating specifications of the system. This value is always lower than the calculated value and if this acceleration value is not high enough then a different motor/amplifier with more power or bandwidth should be utilized.

The PC1x can control either current mode or voltage mode amplifiers. The servo update rate of the PC1x is 488 μ s, for four axes. High "Following Error" can be compensated for using the feedforward coefficients explained

later in this section. There are some general formulas that have been developed to determine acceptable following error for both current and velocity mode systems:

Current mode "Following Error" for $KP = (3^\circ/360^\circ) \times (\text{counts per revolution})$
 Voltage mode "Following Error" for $KP = (90^\circ/360^\circ) \times (\text{counts per revolution})$

It is obvious that the voltage mode allows for much greater following errors than the current mode. This value is the following error when the motor is at peak velocity and will be used when determining the proportional gain (KP).

The "Following Error" for the integral term (KI) or long-term gain value will follow the guidelines below:

Current Mode following error for $KI = 0$ counts
 Voltage Mode following error for $KI = 80^\circ$ of 360° (expressed in motor counts)

1. While still in open-loop mode, hold off (HF), use the DZ command to zero the motor. This variable is used to provide a constant output that will compensate for any torque offset from the load. So, when the system should be stationary, the necessary voltage will be sent to the amplifier to cause the motor to maintain position. With the correct DZ value, the motor should successfully maintain a zero position.

KO is the offset coefficient used while in closed-loop mode, hold on (HN). KO is essentially the same as DZ, but used for closed-loop operation. Once you have determined the correct value for DZ, this same value should be used for the KO variable before beginning to tune the PID filter.

The values for DZ and KO range from -32640 to 32640.

2. Set the known values for velocity, acceleration and the move distance for a trapezoidal profile with at least a 20% flat spot at peak velocity. Formula:

$$\text{Profile distance} = ((\text{peak velocity})^2 / (2 \times \text{acceleration})) \times 2.4$$

$$\text{Example: } ((50,000)^2 / (2 \times 500,000)) \times 2.4 = 6,000$$

3. Execute the move by sending the move commands to the PCIX.

Example: MR6000 GO

4. Adjust the KP term while repeating step 3 until the following error at the flat spot of the profile is acceptable. If the motor becomes unstable prior to obtaining the optimum KP term then increase the KD term until the motor stabilizes.

Example: LP0 KP3 HN MR6000 GO
 LP0 KP10 HN MR6000 GO
 LP0 KP25 HN MR6000 GO
 LP0 KD100 HN

LP0 KP35 HN MR6000 GO
LP0 KD125 HN

The values in the above example are totally arbitrary and may vary drastically with different systems. The LP0 command is used to set the position error to 0.

The values for KP range from 0 to 249.99.

5. Once the KP term has been obtained, then continue executing the motion while raising the KI term until the long-term following error is acceptable. This error can be measured at the two knees of the motion profile. By increasing the KI term, the response time of your system will increase. The motion profile should have a steeper slope as KI increases.

However, as KI increases the system can also become unstable. When the instability becomes unacceptable, increase the KD parameter. This will increase the dampening on the system's motion profile (therefore reducing oscillation, or "ringing".) Continue adjusting the KI and KD terms until the proper response time is obtained.

The values for KI range from 0 to 249.99.

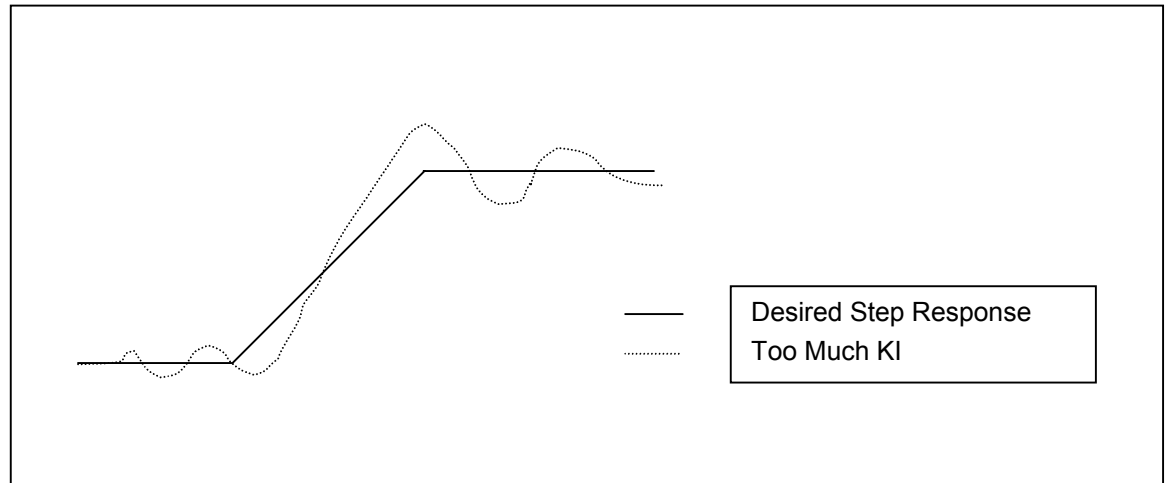


FIGURE 2-7

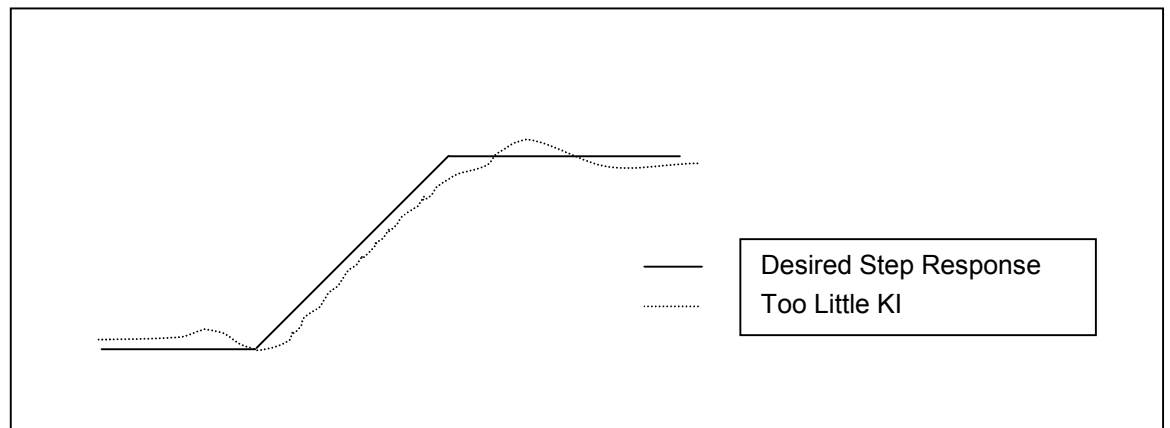


FIGURE 2-8

6. If you are getting too much "ringing" in the motion profile, then increase KD to help dampen the system's response. If, instead, the system is over-damped and is reaching the final velocity too slowly, then reduce the KD parameter. Optimally, the system's motion profile should show the motor reaching the desired velocity as quickly as possible without overshoot and oscillation ("ringing").

The values for KD range from 0 to 249.99.

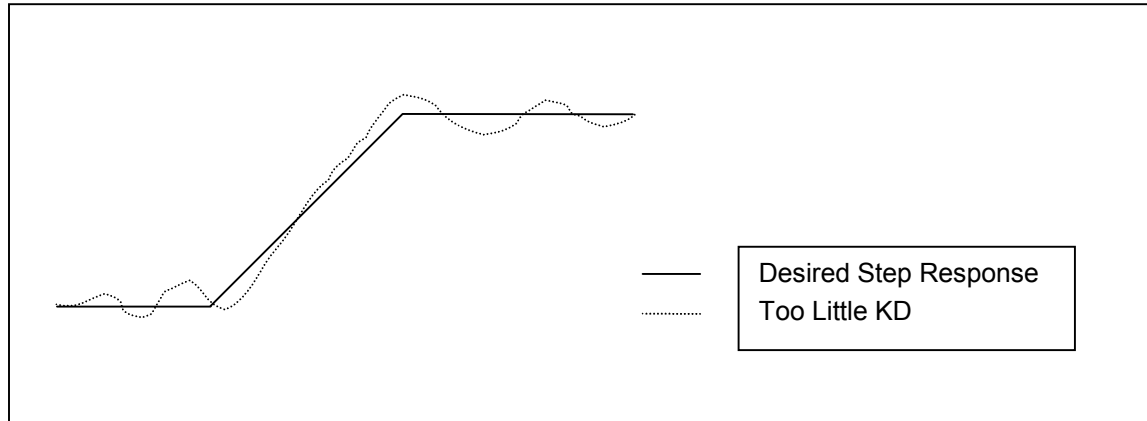


FIGURE 2-9

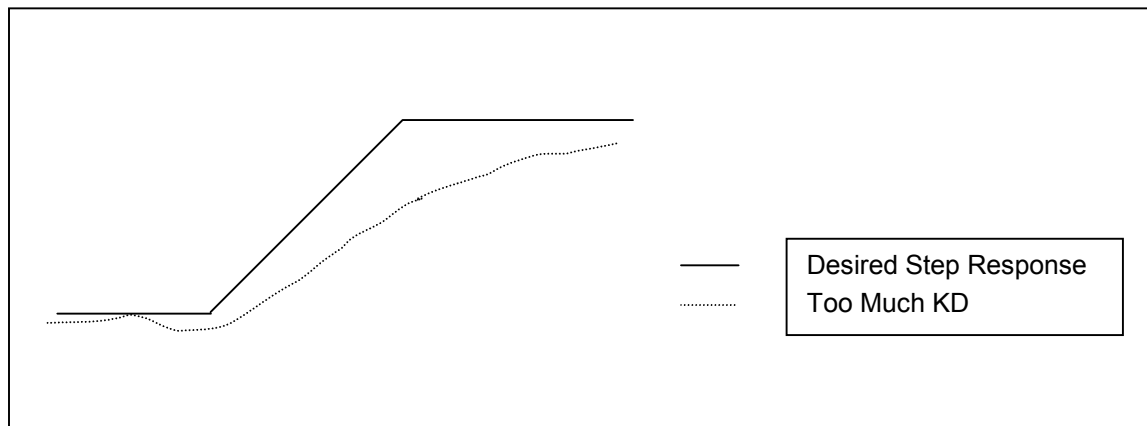


FIGURE 2-10

7. KP, KI, and KD are the primary parameters of concern when tuning a servo system. Once the optimum values for these variables have been determined, then you can adjust some of the secondary parameters that will help fine tune your system's performance. These other variables are described in the subsequent steps.
8. The KV variable is used when tuning velocity controlled servos (voltage mode servo amplifiers.) This is the velocity feedforward coefficient. KV determines how closely the system follows the desired constant velocity portion of the motion profile. By increasing this term, the "Following Error" of the system's response can be minimized. However, too large of a value may result in unstable behavior after command velocity changes.

The values for KV range from 0 to 249.99.

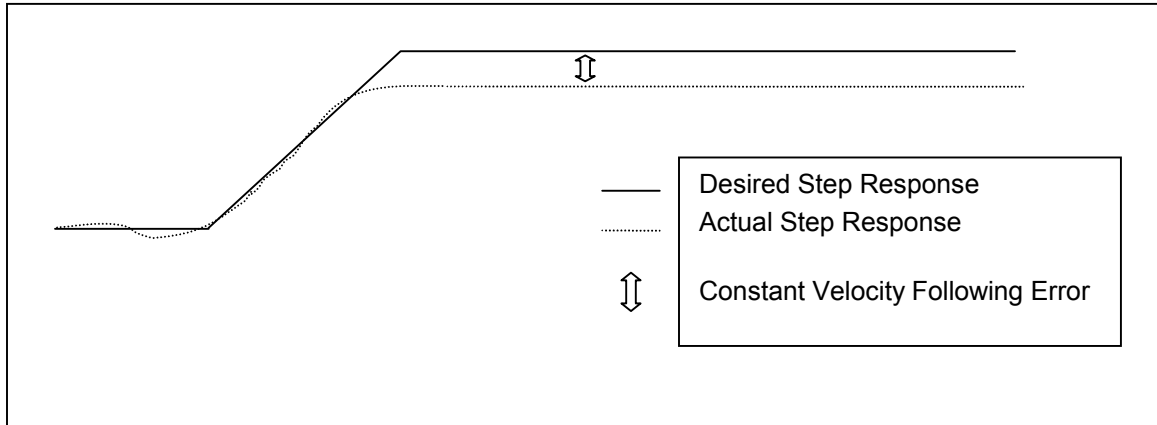


FIGURE 2-11

9. The KA variable is used when tuning torque controlled servos (current mode servo amplifiers.) This is the acceleration feedforward coefficient. Systems with high inertial loads may require additional torque during acceleration or deceleration to achieve optimum performance. KA determines how closely the system follows the desired acceleration and deceleration portions of the motion profile. Increasing this term reduces the following error occurring during acceleration and deceleration of the system. Although, if KA is too large, instability may occur.

The values for KA range from 0 to 249.99.

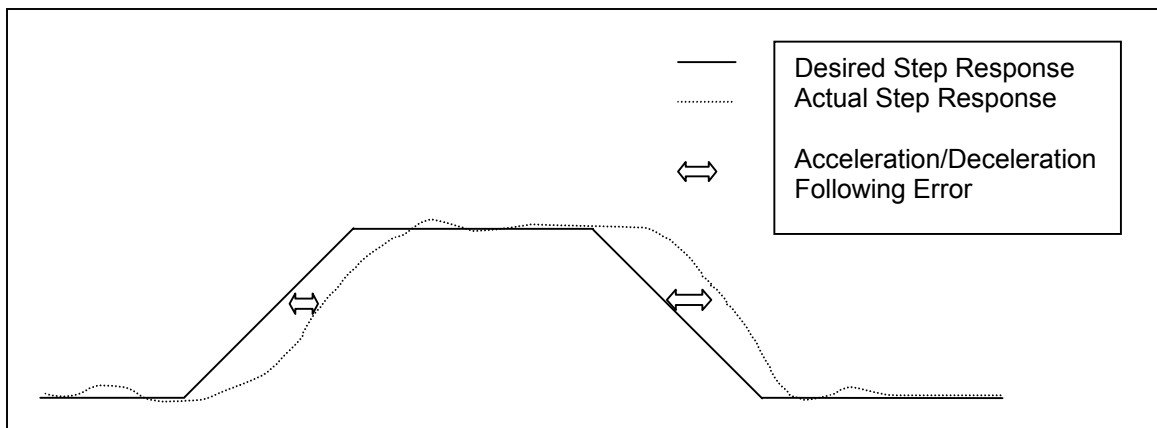


FIGURE 2-12

The block diagram below describes the feedback loop that is taking place in the servo system:

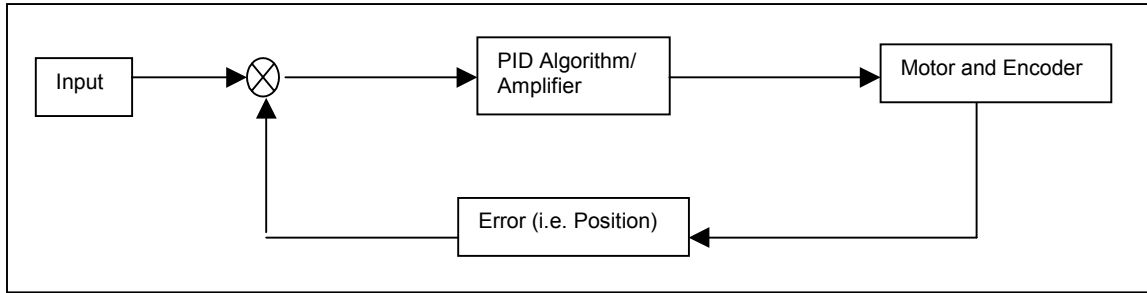


FIGURE 2-13 FEEDBACK LOOP

10. You may want to save the values for KP, KI, KD, etc., for future reference. These values can be saved in the board's Flash Memory, so they can be accessed easily on reset or power-up. The command AP will store your current parameter assignments, such as KP, KI, KD, etc., into flash memory. These saved parameters will then be used as the power up default set of values. Refer to [page 5-10](#) for more detailed information regarding how to use the commands to save and load parameter sets from flash memory.
11. To verify that your motor is tuned properly after you have completed the first 10 steps perform the following test to test the holding torque: Send lp0;HN commands and check the shaft of the motor to make sure it is stiff. If there is play in the motor shaft when you turn it then you may have to re-adjust your PID filter.
12. Once you are satisfied with the static holding torque you could check for position error. Send the command "ac100000;vl5000;mr64000;go". With a 2000 line encoder this move would be equivalent to 8 revolutions of the motor. After the move is complete check the position error by sending the RE and RP commands for the specific axis you are moving. Compare the difference in the two responses. If they are the same then you are on the right track, if the error is greater than 32768 then the controller will disable the PID so that you don't have a runaway motor and major changes to the PID parameters may be required. For minor differences in the encoder and the position reading you can fine-tune your PID filter according to the earlier steps.

NOTE: [Reference Chapter 5 Command Structure](#) for additional information on servo tuning commands.

2.7. SETTING THE USER DEFAULT CONFIGURATION

There are several parameters that can be defined by the user as default. These parameter values can supersede the factory default values and be stored in Flash Memory for power-up configuration. Most of these parameters consist of axis specific values; i.e. velocity, acceleration, limit switch logic sense, etc.

The PC1x comes from the factory with default values for all parameters. For instance, the default value for the velocity of all axes is 100,000 counts per second. (A count is equivalent to a step pulse or one count of an encoder.) In a typical application, when the

system is powered up, the main host computer would initialize all of the peripherals, such as the PC1x, sending to each of the axes the peak velocity. When the User Definable Default Parameter value is defined for the velocity then the initialization of the system can skip initializing the velocities of the defined axes. This feature can greatly simplify the software and initialization process.

Once the values for all of the associated parameters are defined; i.e. velocity, acceleration, PID values, etc. then the “AP” Archive Parameters command is executed to place the values into Flash Memory. From this point forward these defined values will be used after reset or power-up. The individual parameters can be over-written at anytime by using the associated command; i.e. VL#, AC#, etc. To restore the factory defaults the command “RF” Restore Factory defaults is executed. To restore the User Defined Default Parameters the command “RD” Restore Defaults is executed. Refer to [Section 5 Command Structure](#) for more information on these commands.

The following is a list of parameters that can be defined as part of the User Definable Power-Up Default Parameters.

- Overtravel limit (soft limit or hard limit)
- Overtravel limit (enabled or disabled)
- Overtravel limit polarity (active high or active low)
- Software based overtravel for each axis
- Direction Bit polarity
- Acceleration value for each axis
- Trajectory profile for each axis (linear, parabolic, S-curve, custom)
- Velocity Peak
- Velocity Base
- User Unit values for each axis
- Auxiliary output settle time for each axis
- Automatic auxiliary control axis by axis
- Encoder Ratio for each axis
- Encoder Slip tolerance for each axis
- Position Maintenance Dead-Band, Hold Gain and Hold Velocity.
- Servo axis unipolar/bipolar output
- Servo PID values: KP, KD, KI, KO, KV, KA
- Servo zero value: DZ

2.8. POWER SUPPLY REQUIREMENTS

The PC1x Motion Controller Card plugs into the PCI Bus. The PC1x is designed to fit into a standard short card PCI slot, and draws 1.2 Amps from the +5 Volt Power Supply of the PC. For servo models only +12V at 0.1 amp and -12V at 0.1 amp are also taken from the PC.

3.

COMMUNICATION INTERFACE

3.1. INTRODUCTION

The PCIX is 100% compatible with standard PC's and complies with the PCI standards. The PCIX can be considered a motion coprocessor in the PCI computer where it can execute the motion process independent of the PCI CPU or continue to be interactive. The PCI interface of the PCIX has four consecutive 16 bit I/O registers available for control of interrupts, status of interrupt requests, data transfer as well as done flags. The PCIX can generate interrupts for several different conditions, and these are covered in detail in the following sections.

3.2. PCI INTERFACE

The PCI interface to the controller consists of four 16 bit memory mapped I/O registers and 8K bytes (4K words) of dual port RAM.

After the host system BIOS has executed its PCI resource allocation functions the controller's PCI configuration registers will contain the following information:

PCI Register

Offset (HEX) Register Contents (32 bits)

0x000:	The Device ID and Vendor ID. (9050 1065) This register identifies the vendor that supplies the PCI bridge chip.
0x018:	The base address of the controller's dual port RAM.
0x020:	The base address of the controller's memory mapped I/O registers.
0x02C:	The controller's Subsystem ID and Subsystem Vendor ID. (0001 160C) This register identifies the PCI card as an Oregon Micro System controller.
0x03C:	Bits 0 through 3 contain the controller's IRQ assignment.

The dual port RAM is used to pass data from the controller to the host computer. The 16 bit word, at offset zero in the dual port RAM, contains a binary image of the controller's configuration dip switches. Bits 0 thru 2 contain the controller number selection. The

remainder of the RAM is used to pass velocity profile and servo tuning information to the host.

The I/O registers are mapped into the host memory space with offsets from the base address as follows:

Address Offset Register Function

0	Interrupt Control
2	Status
4	Done flags
6	Data

Note bits 8 through 15 of these I/O registers are reserved for future expansion.

3.3. PCI COMMUNICATION THEORY

The process for communicating to the PCIx through the PCI bus at its simplest form consists of the data register and the status register of the PCIx. [Reference Figure 3-1 Data Communication Flow Chart](#) for a flow chart of the communication sequence.

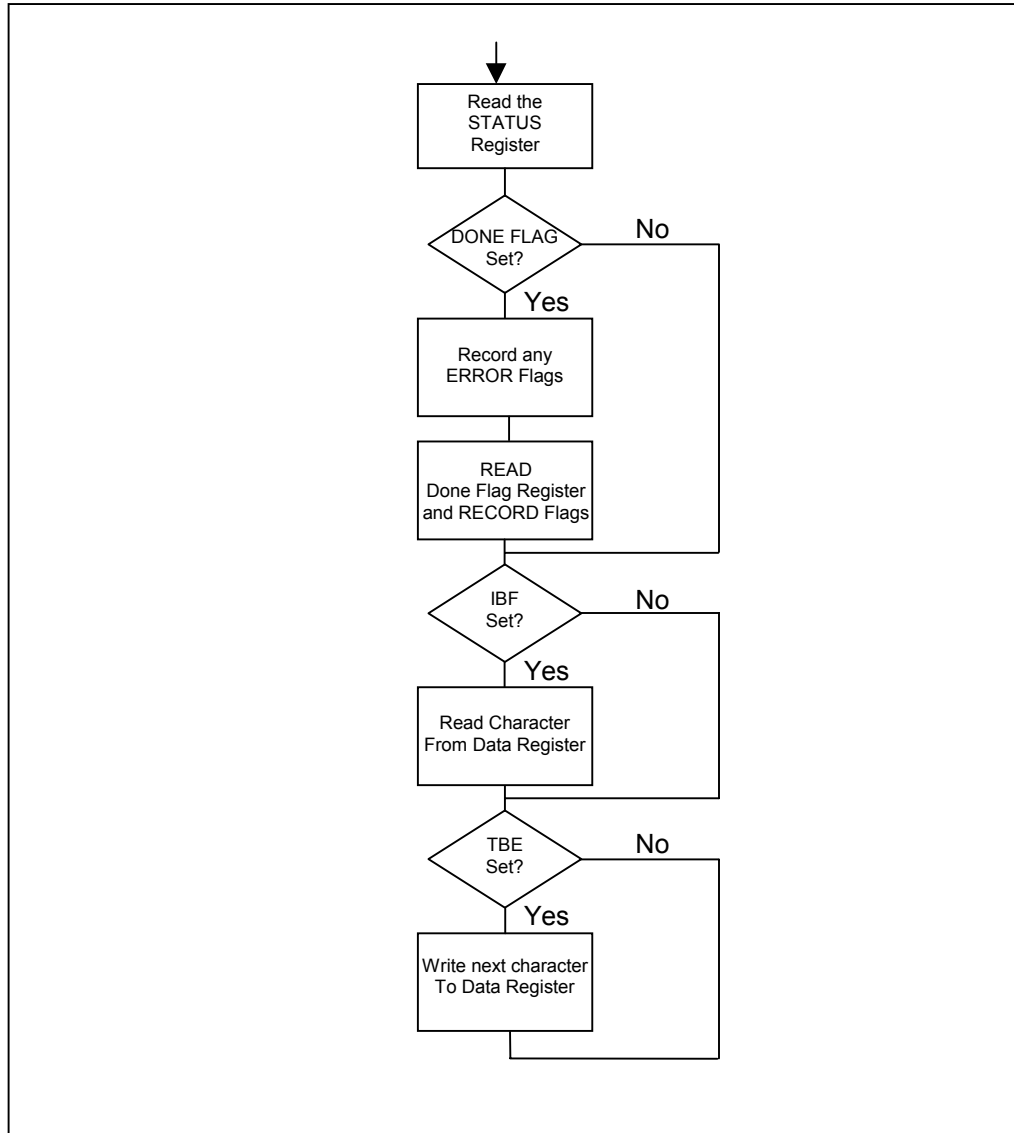


FIGURE 3-1 DATA COMMUNICATION FLOW CHART

The PCIx Master (host) first reads the status register. The information provided in the low order 8 bits informs the host whether the PCIx is ready to receive a character and if there are any characters in the data register to be read by the host. The other information provided in the status registers is not a function of the communication at this point. It is required that the TBE_S bit of the status register is set high before the host can write a character to the data register. The TBE_S bit will go low until the PCIx reads the character

from the data register. The host can either wait for an interrupt generated by a high TBE_S or poll the status register to determine when it can send another character.

When the PCIx sends a character the IBF_S bit will go high. This generates an interrupt to the host. At this point the host would read the data register. This will cause the IBF_S bit to go low until the PCIx sends another character.

3.4. PCI ADDRESS SELECTION

The PCIx is operated as a memory mapped device and occupies a block of 8 bytes of memory addresses. Refer to Table 3-1 Done Flag Register Selection, Table 3-2 Control Register Description, Table 3-3 Status Register Description.

3.5. USING INTERRUPTS

Full interrupt capability is provided by the PCIx in accordance with the PCI specification, but not required. Interrupts provided are: input buffer full (IBF), transmit buffer empty (TBE), overtravel fault (limit), command error and operation complete (done).

3.6. DATA REGISTER

The data register is the data communication port between the PCIx and the PCI Master CPU (host). All data is passed between the PCI Master and the PCIx through this register. The register is full duplex; i.e. it passes data in both directions. This allows for faster processing of the data between the host and the PCIx.

3.7. DONE FLAG REGISTER

The done flag register is a PCI Master read only register of the PCIx. Each bit of this register identifies the done status of each of the axes of the PCIx. It is a 16-bit register where the low 4-bits are used for the four axes of the PCIx. When requested by command, the PCIx will set the appropriate bits of the done flag register to indicate when a process is complete on the associated axes. The host can then read the register at any time to determine the done status of any axes. The register is cleared at the completion of the host read or by the execution of the RA or RI commands. The detail definition of the register is shown in [Table 3-1 Done Flag Register Selection](#).

TABLE 3-1 DONE FLAG REGISTER SELECTION

DONE STATUS REGISTER DESCRIPTION	
BIT	DESCRIPTION
0	Done Status of X Axis
1	Done Status of Y Axis
2	Done Status of Z Axis
3	Done Status of T Axis

3.8. INTERRUPT CONTROL REGISTER

This is a host read/write register that provides different interrupt sources from the PCIx to be individually enabled or disabled. This may be performed at any time by a write to the associated bit of the register. The register may be read back at anytime to verify or determine the state of the interrupts. [Reference Table 3-2 Control Register Description](#) for a detailed definition of the control register.

NOTE: A PCIx Interrupt Service Routine (ISR) should clear the IRQ_E bit on entry and set the IRQ_E bit just before it exits.

TABLE 3-2 CONTROL REGISTER DESCRIPTION

BIT	NAME	CONTROL DESCRIPTION
7	IRQ_E	Interrupt enable bit. This bit must be on to enable any interrupts.
6	TBE_E	Transmit buffer empty interrupt enable bit. This bit should be checked before writing to the data register to avoid sending a character when the interrupt has been disabled.
5	IBF_E	Input buffer full interrupt enable bit.
4	DON_E	Done or error status interrupt enable bit.
3	Unused	
2	Unused	
1	Unused	
0	Unused	

3.9. STATUS REGISTER

This register is a PCI Master (host) read only register that provides status information to the host. This status is independent of the enable status of the interrupt, allowing the board to operate in a polled mode if desired. In an interrupt mode the host would read this register upon receiving an interrupt. The information provided in this register can define the source of the interrupt. The functionality of this and the other registers is consistent with other OMS products. Future expansions to these registers may be performed to enhance the controllers capabilities. See [Table 3-3 Status Register Description](#) for details on the status register.

TABLE 3-3 STATUS REGISTER DESCRIPTION

BIT	NAME	STATUS DESCRIPTION
7	IRQ_S	Interrupt request status.
6	TBE_S	Transmit buffer empty status. This high true bit indicates a character may be written to the transmit buffer.
5	IBF_S	Input buffer full status. This high true bit indicates a character is available in the input buffer
4	DON_S	Done or error status. This high true bit indicates the command is complete; i.e. an ID command has been detected. If bits 0 through 3 are all false it indicates a command completion; i.e. an ID command has been executed. The error bits indicate one or more errors have been detected.
3	OVRT	Overtravel. An overtravel switch was true, indicating attempted travel out of bounds.
2	ENC_S	Encoder error. This bit flags a slip error on models with the encoder option if the interrupt on slip (IS) command has been issued.
1	INIT	Init flag. This bit indicates the PC1x is being reset or the 68332 microprocessor has not completed initialization. Host initialization routines should check this bit for a zero before proceeding.
0	CMD_S	Command error. An unrecognizable command has been detected or LS and LE commands are not in matched pairs.

In order to resolve the source of a done or error interrupt, the DON_S bit (bit 4) of the status register should be read first. This bit in the status register is automatically reset upon the termination of the read cycle. If the DON_S flag is true the error bits should be checked to determine if the interrupt was caused by an error condition. If no error condition is present, the done flag register should be read to determine which axis or axes are done. The Status Register and Done Flag Register clear when read.

The transmit buffer empty (TBE_S) bit of the status register is reset by a host write to the data register and the input buffer full (IBF_S) bit of the status register is reset by a host read of the data register.

4.

CONTROL SIGNAL INTERFACE

4.1. INTRODUCTION

The PCIx family of motion controllers are available in several configurations to manage combinations of servo and step motor systems. The PCI connector incorporates all the control signals of the PCIx. The PCIx controller meets the PCI specification fully and plugs directly into a PCI slot in a computer Motherboard.

4.2. GENERAL PURPOSE I/O, LIMIT AND HOME INPUTS

To facilitate system implementation, limit and home inputs are provided for each axis. Limits may be activated by mechanical switches using contact closures or other suitable active switches, such as a hall effect switch or opto-isolator that connects to ground.

If the motor travels beyond its allowable limits and trips the switch, the limit switch closure removes the excitation from the affected axis. You can select the limit switch active signal state with the LH and LL command on an axis by axis basis. The behavior of the limit functionality can be controlled with the System Control Commands ([see page 5-9 System Status and Control Commands](#)). The PCIx accepts TTL level limit signal inputs.

The home switch provides a means to synchronize the motor controller with the load at some home, or reference, position. The home switch, when used with the software commands HM or HR, causes the motor to decelerate to a stop when the switch closes. On finding the home position, the position counters are initialized to the parameter supplied with the command. You can change the sense of the home switches to TRUE when open by use of the HH command ([described on page 5-50](#)).

The Home switch and general purpose I/O signals to the PCIx are optically isolated on the board. The PCIx expects TTL level signals for the inputs. To insure these inputs operate correctly, the switch input must be connected properly to the PCIx. [Refer to Figure 4-2 Opto Isolated Input Wiring Diagram](#) for a home switch wiring example. (The same connections shown in the diagram apply to a general purpose input bit.)

4.3. CONTROL OUTPUT

The PCIx is configured at the factory for control of servo motors, stepper motors or a combination of both. The servo output may be either unipolar analog (0/+10V) or bipolar analog (-10/+10 V). (see the UN and BI commands in [Section 5 Command Structure \(on page 5-65 and on page 5-64.\)](#))

Step pulse and direction outputs are open-collector TTL level signals which will wire directly into most driver inputs. Refer to Figure 4-1 below.

Auxiliary outputs are optically isolated. Each auxiliary output has a 2.2K Ω pull-up resistor on the PC1x board.

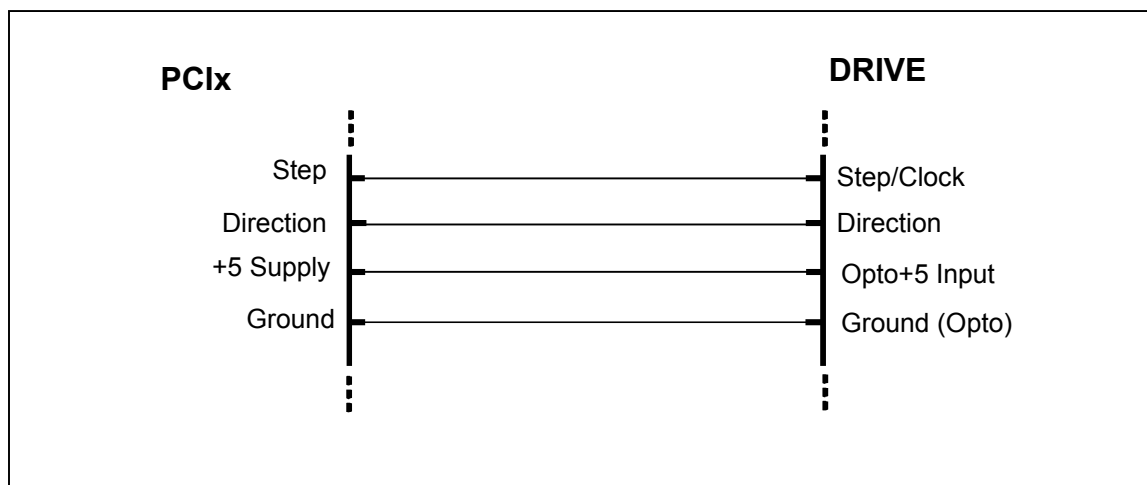


FIGURE 4-1 CONNECTION TO STEP DRIVES

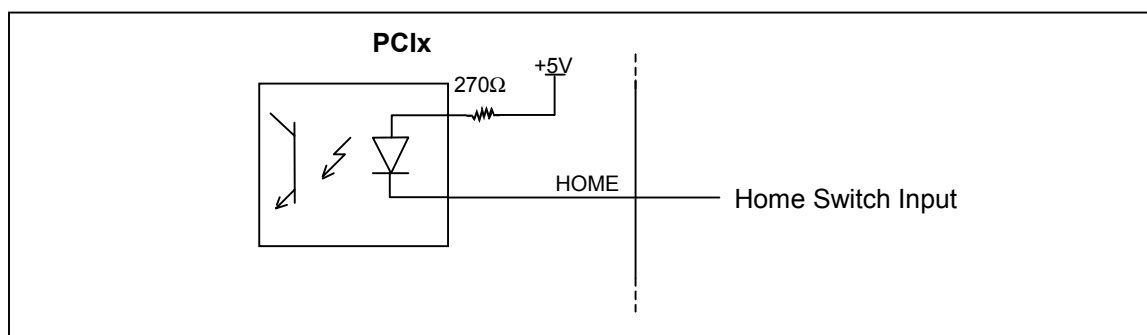


FIGURE 4-2 OPTO ISOLATED INPUT WIRING DIAGRAM

4.4. ENCODER FEEDBACK

Incremental encoder feedback is provided for all servo axes and is optional for the stepper axes. The PC1x encoder feedback accepts quadrature pulse inputs from high resolution encoders at rates up to 4 MHz (after quadrature detection.) The encoder monitors actual position through the encoder pulse train. On servo axes it continuously calculates the position error, amplifies it through the PID filter, and adjusts the output. The stepper axes can monitor the error and correct the position after the move is finished. The encoder input can also be used as an independent feedback source. All modes are capable of slip or stall detection and encoder tracking with electronic gearing. These options are selectable by the user through software commands.

4.5. ENCODER SELECTION AND COMPATIBILITY

The PC1x is compatible with virtually any incremental encoder which provides quadrature outputs. Times four quadrature detection is used to increase resolution. This means that an encoder rated for 1000 counts (or lines) per revolution will result in 4000 counts. The inputs are compatible with encoders, which have single ended or differential TTL outputs. The PC1x inputs have built in hysteresis to minimize effects of noise pickup. The PC1x has differential line receivers to accommodate encoders with differential line driver outputs.

When single ended encoders are used the unused negative inputs; i.e. Phase A-, Phase B-, etc. must be biased at or near +1.5V. This is to be done using the appropriate switch settings on switches S1 and S2. Refer to [Section 2.2 TO PREPARE FOR INSTALLATION](#).

4.6. HOME PROCEDURES

Two logical input functionalities are provided to synchronize the physical hardware with the PC1x controller; i.e. put the controlled motor in the home position.

The home switch input is a TTL level input signal that is optically isolated on the PC1x. A 270 Ω current limiting resistor has been placed in line with the HOME signals on the PC1x. If additional current limiting is required, it should be done externally to the board. Contact Oregon Micro Systems for assistance with this.

The PC1x Home Switch Input can be used to physically home a mechanical stage. When this functionality is used the axis position counter will be reset to a select value when the switch is activated. At this point the PC1x can either ramp the axis to a stop or stop the axis immediately. The control of the direction of travel, the logic active state and the response to the active switch are controlled through commands.

The other homing method on the PC1x uses the Home Switch and the encoder signals to home a motor. When using the HE mode, homing logic is used with these input signals. The home position consists of the logical AND of the encoder index pulse, the home switch input (LOW true only) and a single quadrant from the encoder logic. The home enable pulse must be true for less than one revolution of the encoder thus allowing only one home for the complete travel of the stage. This home input cannot be inverted by the

HH and HL commands while using HOME in the HE mode. The home logic expressed in Boolean terms is:

$$\text{home} = \text{phaseA} * \overline{\text{phaseB}} * \text{index} * \overline{\text{home switch}}$$

It is necessary that the above quadrant occur within the index pulse as provided by the encoder for this logic to function properly. It may be necessary with some encoders to shift the phase of this quadrant by inverting one or both of the phases. Inverting one phase or swapping phase A for phase B will also reverse the direction. The encoder counter (read by an RE command) must increase for positive moves or the system will oscillate due to positive feedback.

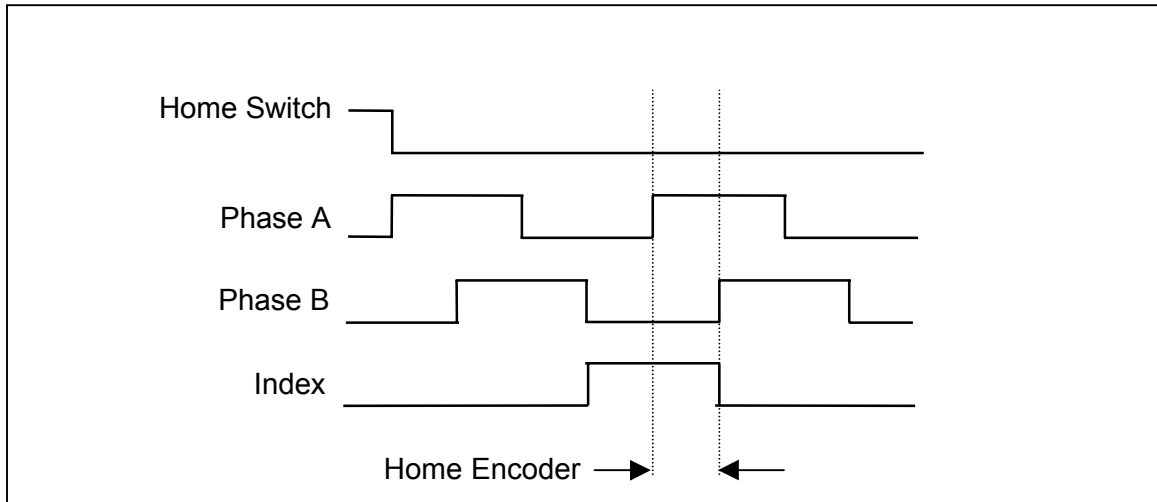


FIGURE 4-3 ENCODER HOMING STATE DETECTION

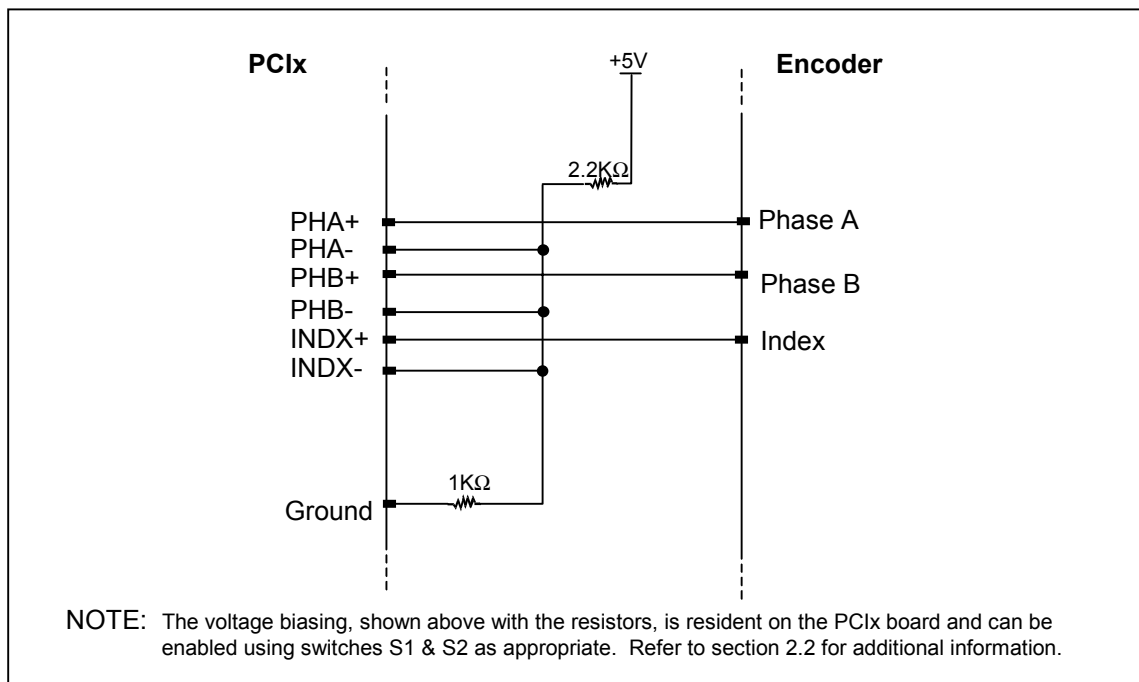


FIGURE 4-4 ENCODER WIRE DIAGRAM FOR SINGLE-ENDED INPUT SIGNALS

TABLE 4-1 OUTPUT CONNECTOR PIN LIST (J2)

SIGNAL CONNECTOR (J2)			
Pin#	Description	Pin#	Description
1	Digital Ground	35	+5VDC
2	I/O-1	36	I/O-0
3	I/O-3	37	I/O-2
4	I/O-5	38	I/O-4
5	I/O-7	39	I/O-6
6	Digital Ground	40	+5VDC
7	X-Index +	41	X-Servo
8	X-Index -	42	X-Step
9	X-Phase A +	43	X-Auxiliary
10	X-Phase A -	44	X-Direction
11	X-Phase B +	45	X-Limit +
12	X-Phase B -	46	X-Limit -
13	Y-Servo	47	X-Home
14	Y-Index +	48	Y-Step
15	Y-Index -	49	Y-Auxiliary
16	Y-Phase A +	50	Y-Direction
17	Y-Phase A -	51	Y-Limit +
18	Y-Phase B +	52	Y-Limit -
19	Y-Phase B -	53	Y-Home
20	Analog Ground	54	+5VDC
21	Z-Index +	55	Z-Servo
22	Z-Index -	56	Z-Step
23	Z-Phase A +	57	Z-Auxiliary
24	Z-Phase A -	58	Z-Direction
25	Z-Phase B +	59	Z-Limit +
26	Z-Phase B -	60	Z-Limit -
27	T-Servo	61	Z-Home
28	T-Index +	62	T-Step
29	T-Index -	63	T-Auxiliary
30	T-Phase A +	64	T-Direction
31	T-Phase A -	65	T-Limit +
32	T-Phase B +	66	T-Limit -
33	T-Phase B -	67	T-Home
34	Digital Ground	68	+5VDC

4.7. IO68-M ADAPTER MODULE

The optional IO68-M is an adapter module designed to provide easy connection for each signal of the PC1x. It incorporates a three row terminal block and some on board filters for the limit inputs. A cable is available with the mating connector to fit the PC1x connector (J2). The +5VDC on the IO68-M is supplied by the PC1x and is protected by a resettable fuse on the PC1x.

This supply voltage is intended to be utilized with accessories used in conjunction with the PC1x such as sensors, motor driver modules, etc., and is specified to supply a maximum current of 0.5 amps for these purposes.

If the fuse detects an over current situation (such as an external short circuit), the supply will shut down. It can be re-activated by powering down the PC1x, ensuring the over current situation has been removed, and by powering the PC1x back up again after 3 seconds.

As the fuse is a semiconductor device, it never has to be replaced and requires no maintenance.

4.7.1. EXPLANATION OF ADDITIONAL CIRCUITRY ON THE IO68-M

The IO68-M board is to be used in conjunction with the PC1x board. The IO68-M board contains low-pass filtering circuitry for the positive and negative limit signals. The default values for the RC constants are 100Ω and $1.0\mu\text{F}$. With these RC constants used the average cutoff frequency is 21.5kHz in a worse case scenario where all other signals are left unterminated.

In addition to the above mentioned circuitry, there are also 2.2k pull-up resistors that have been added to the step output signals for each axis. These resistors have been added to the IO68-M to help reduce noise that may occur on the step output signals.

These resistors and capacitors are packaged as through hole devices. Should your particular application require different RC constants, these devices can be removed from the board and replaced with the appropriate components. When changing these components, use appropriate methods to desolder and solder the components to avoid causing damage to the board (i.e. lifting pads from the board.)

There are two switches on the IO68-M (S43 and S45) that are used in regards to biasing encoder signals. Since the PC1x board has on board switches for encoder signal biasing, the switches on the IO68-M are not needed. So, ALWAYS keep switches 1-6 on S43 & S45 of the IO68-M in the OFF position.

TABLE 4-2

IO68-M SWITCH (S43)

Switch Number	Signal Description
1	(Leave OFF)
2	(Leave OFF)
3	(Leave OFF)
4	(Leave OFF)
5	(Leave OFF)
6	(Leave OFF)
7	(Leave ON)
8	(Leave ON)

IO68-M SWITCH (S45)

Switch Number	Signal Description
1	(Leave OFF)
2	(Leave OFF)
3	(Leave OFF)
4	(Leave OFF)
5	(Leave OFF)
6	(Leave OFF)
7	(Leave OFF)
8	(Leave OFF)

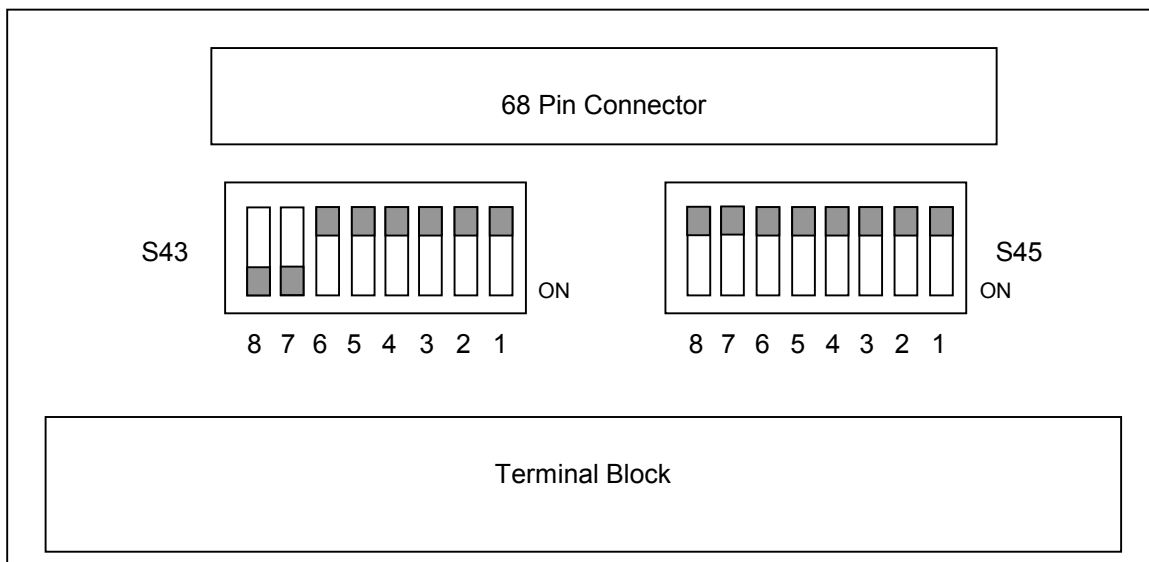


FIGURE 4-5 IO68-M DEFAULT SWITCH SETTING

Switches 7 & 8 of S43 & S45 are used to configure the IO68-M. When the IO68-M is used with the PCIX the switches should be set as shown in [Figure 4-5](#). For the IO68-M, S43, pins 7 and 8 are to be ON, while S45, pins 7 and 8 are to be OFF.

All other signals on the IO68-M are straight through connections, with no additional circuitry added. Should you need filtering circuitry on any of these other signals, it would have to be added external to the IO68-M. Contact Oregon Micro Systems technical support for further instruction.

Table 4-3 shows the pin definitions on the IO68-M terminal block connector.

TABLE 4-3 IO68-M TERMINAL BLOCK PIN-OUT

Row 1	Description	Row 2	Description	Row 3	Description
1	X-Step	24	X-Direction	47	X-Auxiliary
2	X-Phase A+	25	X-Phase B+	48	X-Index +
3	X-Phase A-	26	X-Phase B-	49	X-Index -
4	X-Limit +	27	X-Limit-	50	X-Home
5	+5VDC	28	X-Servo	51	Digital Ground
6	Y-Step	29	Y-Direction	52	Y-Auxiliary
7	Y-Phase A+	30	Y-Phase B+	53	Y-Index +
8	Y-Phase A-	31	Y-Phase B-	54	Y-Index -
9	Y-Limit +	32	Y-Limit-	55	Y-Home
10	+5VDC	33	Y-Servo	56	Digital Ground
11	I/O-0	34	I/O-3	57	I/O-5
12	I/O-1	35	No Connect	58	I/O-6
13	I/O-2	36	I/O-4	59	I/O-7
14	+5VDC	37	Z-Servo	60	Analog Ground
15	Z-Step	38	Z-Direction	61	Z-Auxiliary
16	Z-Phase A+	39	Z-Phase B+	62	Z-Index +
17	Z-Phase A-	40	Z-Phase B-	63	Z-Index -
18	Z-Limit +	41	Z-Limit-	64	Z-Home
19	+5VDC	42	T-Servo	65	T-Auxiliary
20	T-Step	43	T-Direction	66	T-Index +
21	T-Phase A+	44	T-Phase B+	67	T-Index -
22	T-Phase A-	45	T-Phase B-	68	T-Home
23	T-Limit +	46	T-Limit-	69	Digital Ground

This page intentionally left blank

5.

COMMAND STRUCTURE

5.1. INTRODUCTION

An extensive command structure is built into the PCIx family of intelligent motor controls. It includes a 800 command and parameter buffer for each axis and a command loop counter which allows multiple executions of almost any command string.

The following commands in this section are included in the PCIx family of controllers. Most all of the commands consist of two ASCII characters and may be in upper or lower case. Some of the commands expect a numerical operand to follow. These commands are identified with a “#” after the command. *The operand must be terminated by a space, carriage return or semi-colon to indicate the end of the number.* No terminator is required on the other commands, but may be included to improve readability. The operand must immediately follow the command with no space or separation character. The “#” indicates a signed integer input parameter or a signed fixed point number of the format ##.# when user units are enabled. With user units enabled, distances, velocity and acceleration parameters may be entered in units such as inches, revolutions, etc.

Most commands are usable in both single-axis and multi-axis modes. Those that require a numeric parameter in single-axis modes require multiple numeric parameters of the same type in multi-axis modes. For example, the MR (Move Relative) command take a distance as a numeric parameter and is formatted as MR#; in single-axis modes. Multi-axis modes have a parameter position for each axis and must be formatted as MR#,#,#,#; in a 4-axis system. (Note use of commas (,) between axes.) Any # parameter may be omitted for any axis which is not to be affected by the command and the command may be terminated prematurely with a semicolon. For example, to move only the Y and Z axes, enter the command as MR,#,#;.

Some commands that are usable in both single-axis and multi-axis modes do not take a parameter in single-axis modes. These commands still require numeric parameters in multi-axis modes but the parameters take on a Boolean meaning. These commands are identified by their formatting as “XXb,b,b,b;” where XX is the command name and the ‘b,b,b,b’ represents a set of Boolean values. A Boolean “true” is any value greater than zero. A Boolean “false” is any absent or zero value. For example, the single-axis format of LN (Limits ON) is simply LN without parameters of any kind. The multi-axis format of LN is LNb,b,b,b; for 4-axis systems where each ‘b’ represents the Boolean value for the corresponding axis. Like other multi-axis commands, a ‘b’ may be omitted if that axis is to remain unchanged and the command may be prematurely terminated with a semicolon. Each ‘b’ position, if used, is expected to be any value greater than zero. For example, to enable the Y and Z axes limit switches and leave X and T unchanged, send the command “LN,1,100;”. The 1 and 100 may actually be any value greater than zero. If a zero is used as a parameter, the effect will be the same as if the value were omitted completely; i.e. LN0,0,1; is the same as LN,,1;.

Synchronized moves may be made by entering the AA or AM command. These commands perform a context switch which allows entering motion commands in the format MRx#,y#,z#,t#;. Numbers are entered for each axis which is to be commanded to move. An axis may be skipped by entering a comma with no parameter. The command may be prematurely terminated with a ";", i.e. a move requiring only the X and Y axes would use the command MRx#,y#; followed by the GO command. Each axis programmed to move will start together upon executing the GO command. The PCIX can be switched back to the unsynchronized mode by entering the desired single axis command such as AX.

The AM command is provided for complex applications where the host manages multiple motion processes by a multitasking operating system. This mode shares the same instructions as the AA mode, but allows starting a task while some other task involving one or many axes is active. For example, the X and Y axes could be doing linear interpolation while the Z axis is making an unrelated move simultaneously.

Constant velocity contouring provides another mode wherein the move parameters are predefined by entering AA (or AM) then CD#,#;. The PCIX will then calculate the move profile in advance and move at constant velocity in the prescribed pattern. It can do linear interpolation on as many as 4 axes between the predefined points or it can do circular interpolation mixed with linear on two axes.

5.2. COMMAND QUEUES

The input characters are placed in a character buffer on input then removed and interpreted. The commands are then placed in separate command queues for each axis. As they are executed the space is reclaimed allowing the host to pass commands ahead of the moves actually being processed. Most of the commands are placed in the appropriate command queue for execution, while others are executed immediately allowing return of status information in a timely way rather than when encountered in the command stream. This information is provided in a table for each command which shows the queue requirements, if any, and indicates immediate in those cases where the command is not queued. The queue requirements shown in the tables are typical. Depending on the circumstances in which the command is issued, the actual queue requirement may vary slightly. The single axis cases are indicated by the mode reference indicating the appropriate axis. The synchronized mode is indicated by the mode identifier AA or AM. The contouring case is indicated by AA/CD for multiple axes in contour definition mode. The RQ command may be used to determine the actual space available at any time. The queues operate independently allowing each axis to perform separate processes concurrently. The synchronized modes (AA) insert special wait opcodes which allow the axes to be synchronized in this mode. When the commands are nested within loops, the queue space is not reclaimed until after the loop has been executed the programmed number of times. For loops larger than the queue space, the loop may never be completed since it cannot reclaim the queue space and cannot accept the loop terminator. The RQ command may be used to examine the remaining queue space. A "CTRL-D" may clear this condition if the input character queue is not also filled since it bypasses the command interrupter.

Some commands are valid only for stepper axes, others for stepper axes with encoder feedback, and still others for servo axes. Most are valid for all three types or some combination of types. A set of symbols to the right of each command identifies which motor types with which each command may be used. The symbols' meanings are as follows:

 Stepper motor with or without an encoder

 Stepper motor with an encoder

 Servo motor

If a command is usable with one of these motor types, the symbol will appear in black. If the command is not usable with a motor type, that motor symbol will be displayed in gray:

 This command is not usable with servo motors

 Indicates an example.

The following commands are available in firmware revision 1.00 and above.

5.3. COMMAND SUMMARY

The following commands are included in the PC1x family of motor controllers. The '#' indicates a signed integer input parameter or a signed fixed point number of the format ##.# when user units are enabled. With User Units enabled, distances, velocity and acceleration parameters may be input in inches, revolutions, etc.

COMMANDS IN CHAPTER 5		
COMMAND	SECTION PAGE NUMBER	COMMAND DESCRIPTION
AA	5-16	Any following commands are for the AA (All Axes) mode
?AC	5-125	Report AC command
AC	5-124	Acceleration, set acceleration/deceleration register
?AD	5-45	Report default auxiliary bit state
ADH	5-43	Set auxiliary default to high
ADL	5-44	Set auxiliary default to low
AF	5-40, 5-181	Auxiliary off
AM	5-17	Axes multitasking mode
AN	5-39, 5-182	Auxiliary on
AP	5-10	Make the current parameter set the power up default values
AT	5-21	Any following commands are for the T axis
AX	5-18	Any following commands are for the X axis (default on reset)
AY	5-19	Any following commands are for the Y axis
AZ	5-20	Any following commands are for the Z axis
BH	5-47	Set selected I/O bit high (off)
BI	5-64	Bipolar, set the analog torque outputs to bipolar

COMMANDS IN CHAPTER 5		
COMMAND	SECTION PAGE NUMBER	COMMAND DESCRIPTION
BL	5-46	Set selected I/O bit low (on)
BS	5-48	Set all bits of the general purpose output port to the state specified by the hex argument.
BW	5-190	Wait for input to go low
BX	5-49	Return bit status in hex format
CA	5-167	Clear done flag of currently addressed axis
CD	5-177	Define a contour
CE	5-183	End contour definition, ramp to a stop
CK	5-184	End contour definition, immediately stop step pulses
CN	5-132	Cosine on, enable cosine velocity profiles
CR	5-179	Circular interpolation, move in a circle
CV	5-175	Contouring velocity, definition
CW	5-199	Clear while flag; i.e. terminate WH/WG loop
CX	5-185	Contour execute
?DA	5-139	Print a custom ramp
DAB	5-137	Define custom ramp breakpoint
DAE	5-138	End custom ramp definition
DAR	5-136	Begin custom ramp definition
?DB	5-76	Report direction bit logic
?DE	5-141	Report an acceleration ramp definition table entry
?DS	5-140	Report the size of a custom acceleration ramp table
DBI	5-74	Invert direction bit
DBN	5-75	Normalize direction bit
DC	5-127	Set the deceleration rate that will be used by the GU command
?DZ	5-71	Report DAC open-loop offset
DZ	5-70	Offset coefficient, used in open-loop mode
EA	5-31	Encoder status, return encoder status of currently addressed axis
EF	5-15	Echo off, turn off echo to host (default at power up)
EN	5-14	Echo on, turn on echo to host
?ER	5-92	Report motor:encoder ratio
ER	5-90	Encoder ratio, set encoder count to motor count ratio
?ES	5-100	Report encoder slip tolerance
ES	5-99	Encoder slip tolerance, set tolerance before slip or stall is flagged
ET	5-105	Encoder tracking, set encoder tracking mode
FL	5-23	Flush an axis command queue
FP	5-173	Force position, flush queue and attempt to stop at specified position
GD	5-152	Go and reset done flags
GO	5-151	Go command, start execution of motion

COMMANDS IN CHAPTER 5		
COMMAND	SECTION PAGE NUMBER	COMMAND DESCRIPTION
GS	5-153	Go and use the home switch to monitor for motor slip
GU	5-154	Go and use the AC values to accelerate and the DC values to decelerate
?HD	5-94	Report position maintenance deadband
HD	5-93	Hold deadband, specify deadband tolerance for position hold
HE	5-52	Encoder home mode, set home on encoder logic
HF	5-88	Hold off, disable position hold, slip detection and tracking modes
?HG	5-98	Report position maintenance gain
HG	5-97	Hold gain, specify position hold gain parameter
HH	5-50	Home high, home switches are active high
HL	5-51	Home low, home switches are active low
HM	5-106	Home, find home and initialize the position counter
HN	5-87	Hold on, enable position correction after move
HR	5-108	Home reverse, find home in reverse direction and initialize position counter
HS	5-53	Home switch, enable home switch mode
?HV	5-96	Report position maintenance velocity
HV	5-95	Hold velocity, specify maximum position hold correction velocity
IC	5-166	Interrupt clear, clear done interrupt status and error flags
ID	5-162	Interrupt host when done and set done flag
II	5-163	Interrupt independent
IN	5-164	Interrupt when nearly done
IP	5-165	Interrupt when in position
IS	5-101	Interrupt slip, interrupts host on slip or stall detection
JF	5-144	Jog at fractional rates
JG	5-142	Jog command, run motor at specified velocity until a new velocity command is sent or it is stopped by a stop or kill command
?KA	5-78	Report acceleration feed-forward
KA	5-77	Acceleration Feedforward coefficient, used in tuning servo systems
?KD	5-86	Report PID derivative gain
KD	5-85	Derivative Gain coefficient, used for PID filter
?KI	5-84	Report PID integral gain
KI	5-83	Integral Gain coefficient, used for PID filter
KL	5-160	Kill, flush queue and terminate pulse generation immediately on all axes without decelerating
KM	5-109	Home and kill pulse generation
?KO	5-73	Report PID closed- loop offset
KO	5-72	Offset coefficient, used in closed-loop mode
?KP	5-82	Report PID proportional gain
KP	5-81	Proportional Gain coefficient, used for PID filter
KR	5-110	Home in reverse and kill pulse generation

COMMANDS IN CHAPTER 5		
COMMAND	SECTION PAGE NUMBER	COMMAND DESCRIPTION
KS	5-161	Kill selected axes
?KV	5-80	Report velocity feedforward
KV	5-79	Velocity Feedforward coefficient, used in tuning servo systems
LA	5-129	Linear ramp selection per axis
LE	5-193	Loop end, terminate most recent LS command
LF	5-54	Disable limit switches for selected axis
LH	5-56	Limit high, limit switch is active high
LL	5-57	Limit low, limit switch is active low
LN	5-55	Enable limit switches for selected axis
LO	5-112	Set axis motor position but not the encoder position
LP	5-111	Load position, load position counter with parameter
?LS	5-58	Report limit active state
LS	5-191	Loop start, set loop counter, from 1 to 32000 loops; (may be nested to 4 levels)
MA	5-147	Move absolute, move to absolute position
MD	5-32	Define a temporary macro
ML	5-149	Move linear, move specified distance relative from current position
MM	5-169	Move minus, set minus direction for MV type move
MO	5-146	Move one pulse in current direction
MP	5-168	Move plus, set positive direction for MV type move
MR	5-148	Move relative, move specified distance from current position
MT	5-150, 5-180	Move to, move to specified position
MV	5-170	Move velocity, move to first parameter (absolute position) at second parameter velocity without stopping at end of move
MX	5-34	Execute a macro command string
NV	5-176	Set a new contour velocity from within a contour
?PA	5-38	Report power automatic state
PA	5-36	Power automatic, turn power on before each move and off after the move
PE	5-118	Report encoder positions of all encoder and servo axes
PF	5-128	Parabolic off, disable parabolic ramps; i.e. linear ramps will be generated
?PM	5-89	Report PID state
PM	5-35	Print a macro command string
PN	5-130	Parabolic on, enable parabolic ramps
PP	5-115	Report motor positions of all axes
PR	5-131	Parabolic ramp selection per axis
PT	5-33	Preserve a temporary macro by copying it to non-volatile memory
QA	5-27	Query status of switches and flags for addressed axis without affecting flags
QI	5-28	Query status of switches and flags on all axes without affecting flags
RA	5-29	Return status of switches and flags and reset flags

COMMANDS IN CHAPTER 5		
COMMAND	SECTION PAGE NUMBER	COMMAND DESCRIPTION
RC	5-126	Return current acceleration or deceleration of the current axis
RD	5-11	Restore the current parameter set to the power up default values
RE	5-117	Request encoder position, return current encoder position
RF	5-12	Restore the current parameter set to the factory default values
RI	5-30	Return status of switches and flags for all axes and reset flags
RL	5-102	Return slip status of each axis
RM	5-113	Return remainder of position divided by parameter in position counter
RP	5-114	Request position, returns current position
RQ	5-22	Request queue status, return number of queue entries available
RS	5-13	Software reset of PCIx
?RT	5-135	Report ramp type
RU	5-116	Return current position in user units
RV	5-121	Return current velocity at which the axis is moving
SA	5-156	Stop all, flush queue and stops all axes with deceleration
SC	5-133	Cosine ramp selection per axis
SD	5-157	Stop all axes and clear any done flags
?SE	5-42	Report settling time
SE	5-41	Set settling time before power is reduced in PA mode
SF	5-60	Soft limit off, restore normal overtravel operation
SI	5-158	Stop selected motors
?SL	5-61	Report soft limit status
SL	5-59	Soft limit mode, allow pulse train to ramp down on overtravel
?SO	5-66	Report analog output mode
SO	5-159	Stop at a designated position using a specified ramp down distance
SP	5-172	Stop at position, stop at specified position if possible after all commands have been executed
SR	5-134	Select a custom acceleration ramp for use by the currently active axis
ST	5-155	Stop, flush queue and decelerate to stop
?SV	5-69	Report servo voltage invert selection
SVI	5-67	Invert servo voltage
SVN	5-68	Normalize servo voltage
SW	5-189	Sync wait, wait for the input bit to be released (non-active)
TF	5-104	Turn encoder slip kill off
TM	5-145	Jog at the current velocity for the specified number of milliseconds
?TL	5-63	Report software overtravel ranges

COMMANDS IN CHAPTER 5		
COMMAND	SECTION PAGE NUMBER	COMMAND DESCRIPTION
TL	5-62	Set software travel limits
TN	5-103	Turn encoder slip kill on
UF	5-25	User units off, turn off user unit translation
UN	5-65	Unipolar, set the analog torque outputs to unipolar
?UU	5-26	Report axis user units' axis assignment
UU	5-24	User units, multiply acceleration, velocity and distance parameters by specified parameter
?VB	5-123	Report axis base velocity
VB	5-122	Base velocity, set base velocity
?VL	5-120	Report axis velocity limit
VL	5-119	Set maximum velocity to be used in profile
VS	5-174	Velocity stream, slave velocity mode for profiling
WA	5-187	Wait until all moves on all axes are finished
WD	5-195	While end, WS loop terminator
WG	5-198	Terminate WH loop
WH	5-196	While, execute all commands until WG loop terminator, until flag cleared by CW command
WQ	5-188	Wait until current axis queue is empty
WS	5-194	While sync, execute while sync of input is true
WT	5-186	Wait, wait for specified number of milliseconds
WY	5-9	Who are you , returns model and software revision

5.4. SYSTEM STATUS AND CONTROL COMMANDS

These commands allow the host to request the status of various move parameters, including the status of limit and home switches and/or allow control of various system parameters and operating modes to allow the user to optimize the response of the system for the application.

5.4.1. IDENTIFICATION COMMANDS

WY WHO ARE YOU



The WY command returns the model type, firmware revision number, and number of controlled axes of the board being addressed. The response string will be delimited with linefeed+carriage return pairs.



Example: You want to examine the board identification string.

Enter: WY

Response: <LF><CR>PCIX ver 1.00-004 S/N 001234<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
WY	AX – AT	Immediate	
WY	AA-AM	Immediate	
-	AA/CD	Not Valid	

5.4.2. POWER-UP DEFAULTS

The PCIX can store most user-settable parameters and reload them when the board powers-up or is reset. The following commands can be used to store these parameters, return the board to factory default, reload the stored parameters, and reset the board to reload the stored parameters.

AP ASSIGN CURRENT PARAMETERS AS POWER-UP DEFAULTS



The AP command stores the current parameter set as the power-up default set of values. This is done by writing the current parameter set to flash memory. The following list of parameters will have their values saved to flash memory when the AP command is used: SF/SL, LN/LF, LH/LL, AC, LA/SC/PR/SC, VL, VB, UU, SE, PA0/PA1, ER, ES, HD, HV, HG, BI/UN, KP, KD, KI, KO, KV, KA, and DZ.



Note: This command should not be issued when an axis is in motion and it should be used sparingly because the flash memory has a limited number of write cycles.



Example: Save the current parameter set to be the power up default set of values.

Enter: AP

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
AP	AX – AT	Immediate	
AP	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: RD, RF

RD RESTORE DEFAULT VALUES

The RD command restores the current parameter settings of the PCIX to the power-up defaults. These defaults will be user-set parameters if they have been previously archived using the AP command. Otherwise, the factory defaults will be used.



Example: Assign the current parameter set to be the default values.

Enter: RD

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
RD	AX – AT	Immediate	
RD	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: AP, RF

RF RESTORE FACTORY DEFAULT VALUES



RF restores the current parameter set to the factory default values. This process does not alter the parameters stored in flash via the AP command. To restore the flash memory to factory default, the RF command must be issued followed by the AP command.



Example: Assign the current parameter set to be the factory default values.

Enter: RF

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
RF	AX – AT	Immediate	
RF	AA-AM	Immediate	
-	AA/CD	Not Valid	

FACTORY DEFAULT SETTINGS
Axis direction bit output state is normal(DBN)
Overtravel limits are active low(LL)
Soft limits are disabled(SF)
Acceleration is 2000000 (AC2000000)
Use linear acceleration ramps(LA)
Early deceleration factor is zero(ED0)
Velocity is 200000 (VL200000)
Base Velocities zero(VB0)
User units are off(UF)
Auxiliary output bits settle times are zero(SE0)
Auxiliary output bits power control is disabled(AN)
Auxiliary output bit power up states are high(ADH)
Software overtravel limits are disabled(TL0,0;)

ENCODER MODELS
Motor/encoder ratio is one to one(ER1.0)
Encoder slip tolerance is one(ES1)
Position maintenance dead band is zero(HD0)
Position maintenance velocity limit is zero(HV0)
Position maintenance hold gain is one(HG1)

SERVO MODELS
DAC output is bipolar(BI)
DAC zero offset is zero(DZ0)
Servo output voltage is normal(SVN)
PID proportional gain is 10 (KP10.00)
PID differential gain is 20 (KD20.00)
PID integrator gain is 0.04 (KI0.04)
PID acceleration feed forward is zero(KA0.00)
PID velocity feed forward is zero(KV0.00)
PID offset is zero(KO0)

Related commands: AP, RD

RS**RESET**

The RS command is a software reset which causes the local PCIX microprocessor to reset. *This process forces a reset of all components on the PCIX controller and this command should be used sparingly.* All previously entered data and commands are lost. All internal parameters are initialized to defaults. All interrupts are disabled. This command is intended for catastrophic failure recovery only. The KL command should be used to reset queues or return the system to a known state.

Any running interrupt service routine should be halted or paused when issuing this command to avoid unwanted interrupt traffic. If the application is using drivers supplied by OMS, the application should use the reset function in the driver rather than issuing the RS command to avoid unnecessary complications.



NOTE: We recommend limiting the use of this command because of the effect it may have on the PCI interface electronics. This command totally resets all of the electronics on the controller and clears out the configuration registers of the PCI bus logic as well as all others on the controller. This could potentially cause the computer to crash or have other unexpected results.



Example: Reset the PCIX entirely as though it had been power-cycled

Enter: RS

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
RS	AX – AT	Immediate	
RS	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: KL, SA

5.4.3. COMMAND ECHO

Normally the PC1x will not “echo” every character written to its data register; i.e. it will not write the same character back to it’s own data register for the host to read back. The PC1x can; however, echo every single character sent if desired to verify that each character has been received by the board. This will consume more communication bandwidth than without echo so this mode should be used only as a debugging mechanism.

EN**ECHO ON**

The EN command enables echoing. All commands and parameters sent to the PC1x will be echoed to the host. This mode is useful for debugging command strings from a terminal. This mode also outputs an English readable error message (when an error occurs) to the host which may be echoed to the screen to aid in debugging.

**Example:**

Enable echoing from the PC1x so that commands are echoed and error messages are returned to the host as readable ASCII strings. This command would probably be the first command executed after turning on the system when this mode is desired. Note that the EN command itself is not echoed.

Enter:

EN

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
EN	AX – AT	Immediate	
EN	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: EF

EF**ECHO OFF**

The EF command disables echoing from the PCIX (refer to the EN command). This is the default mode at power up or reset.



Example: Stop echoing to the host. Note that the EF command will be echoed.

Enter: EF

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
EF	AX – AT	Immediate	
EF	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: EN

5.4.4. QUEUE SELECTION COMMANDS

The following commands set the context to direct the commands which follow to the appropriate axis. They remain in effect until superseded by another command of the same type, specifying a different axis.

AA
ALL AXES


The AA command performs a context switch to multi-axis mode. All commands entered after this one will be treated as “all axis” commands which must be formatted for multi-axis use rather than single-axis use. Each command will be executed in the order in which it is received. This is true even if the second command affects axes other than those affected by the first command. For example, if AA mode is entered followed by a move of the X axis and then a move of the Y axis, the Y axis move will not begin until the X axis move has completed.



Example: Perform an absolute move using the X and Y axes. When this move is complete, perform a relative move using the Y, Z, and T axes.

Enter: AA MA12000,14000; GO MR,5000,1500,100000; GO

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
AA	AX – AT	2	2
AA	AA-AM	2	2
-	AA/CD	Not Valid	

Related commands: AM, AX, AY, AZ, AT, CD

AM**AXES MULTITASKING**

The AM mode allows several tasks to be managed simultaneously. This command changes the mode of all future commands to multi-axis mode. In this mode, a task may be performing coordination motion on 2 axes, while a second task is performing unrelated but simultaneous motion on another axis. All commands sent in this mode must be formatted for multi-axis mode rather than single-axis mode.



Example: Perform a coordinated relative move on the X and Y axes, while moving the T axis as a separate move at the same time.

Enter: AM ML2000,3000; GO MA,,,10000; GO

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
AM	AX – AT	Immediate	
AM	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: AA, AX,AY, AZ, AT

AX**AXIS X**

The AX command directs all following commands to the X axis. This is the default mode at power up or reset. Commands sent in this mode will be executed in the order in which they are received (subject to queued versus immediate execution) and are expected to be formatted for single-axis use.



Example: Make the X axis step at a rate of 5,000 steps/second.

Enter: AX JG5000;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
AX	AX – AT	Immediate	
AX	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: AA, AM, AY, AZ, AT

AY**AXIS Y**

The AY command directs all following commands to the Y axis. Commands sent in this mode will be executed in the order in which they are received (subject to queued versus immediate execution) and are expected to be formatted for single-axis use.



Example: Examine the status of the Y axis.

Enter: AY RA

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
AY	AX – AT	Immediate	
AY	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: AA, AM, AX, AZ, AT

AZ**AXIS Z**

The AZ command directs all following commands to the Z axis. Commands sent in this mode will be executed in the order in which they are received (subject to queued versus immediate execution) and are expected to be formatted for single-axis use.



Example: Move the Z axis 2,000 steps at a rate of 500 steps/second.

Enter: AZ VL500; MR2000; GO

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
AZ	AX – AT	Immediate	
AZ	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: AA, AM, AX, AY, AT

AT**AXIS T**

The AT command directs all following commands to the T axis. Commands sent in this mode will be executed in the order in which they are received (subject to queued versus immediate execution) and are expected to be formatted for single-axis use.



Example: Move the T axis to absolute position -2468.

Enter: AT MA-2468 ; GO

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
AT	AX – AT	Immediate	
AT	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: AA, AM, AX, AY, AZ

5.4.5. QUEUE STATUS COMMANDS

Commands sent to the PClx are either queued or immediate. As the type names imply, queued commands are stored in first-in-first-out buffers to be executed in the order they were sent while immediate commands are executed the moment they are received.

There are several internal queues in the PClx corresponding to the various axis and command modes and each of these queues has a limited amount of storage space. For example, the X axis queue can hold 800 “units” and the contour queue can hold 7160 “units”. Each command requires some number of storage units; the amount of storage required is listed in a table with each command.

The following commands provide control and monitoring capability for the command queues.

RQ
REPORT QUEUE STATUS


The RQ command returns the number of entries available in the queue of the currently addressed axis in single axis mode or all axes separated by commas in multi-axis modes. The ASCII string is delimited by linefeed + carriage return pairs. The maximum available in each command queue is 800 except the contour queue which is 7160. The response is at a fixed length of 3 characters for all modes except contour mode which is fixed at 4 characters. For example, if the current free queue space is 67 in AM mode, the response from the board to the RQ command is <LF><CR>067<LF><CR>.



Example: See the size of the command queue for the T axis.

Enter: AT RQ

Response: <LF><CR>800<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
RQ	AX – AT	Immediate	
RQ	AA-AM	Immediate	
RQ	AA/CD	Immediate	

Related commands: FL, AX, AY, AZ, AT, AA, AM, CD

FL**FLUSH**

The FL command will flush an individual axis' command queue. This command is similar in operation to the KL and ST commands except that current motion will remain unaffected by the FL command. All unexecuted commands remaining in the current axis queue will be flushed upon receipt of this command.



Example:

Several motion commands have been sent to the X axis but a situation arose and now those commands must be cleared out. The currently executing motion must be allowed to complete to avoid damage to the product.

Enter:

AX FL

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
FL	AX – AT	Immediate	
	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: KL, KS, RS, ST, SA

5.4.6. USER UNIT COMMANDS

The following commands allow specification of move parameters in user defined units. The OMS controls will automatically convert all move parameters to these units once they have been initialized.

UU

USER UNITS



The UU command converts all move velocities, distances, etc. to user specified units by multiplying by the specified parameter. The UF command is used to terminate this mode. Factory default is with this command off. See the AP Command on [page 5-10](#) to preserve the UU settings as the Power up/Reset values.



Example: The motor, driver and gear ratio you are using requires 10,000 steps to move one inch. Set up the X, Y and Z axes so you can enter move information in inches.

Enter: AX UU10000; AY UU10000; AZ UU10000;
 or
 AA UU10000,10000,10000;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
UU#;	AX – AT	Immediate	
UU#,#,#,#;	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: RU, UF, ?UU

UF**USER OFF**

The UF command turns off user units returning all numeric commands and responses to their factory default raw representations. This command is equivalent to and preferred over UU1 since it turns off the mode thus minimizing unnecessary overhead. [See the AP Command on page 5-10](#) to make the UF settings as the Power up/Reset values.



Example: Turn off user unit conversion on the X, Y and Z axes.

Enter: AX UF AY UF AZ UF

or

AA UF1,1,1;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
UF	AX – AT	Immediate	
UFb,b,b,b;	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: UU, ?UU

?UU**REPORT AXIS USER UNITS'
AXIS ASSIGNMENT**

This command returns the current user units' multiplier as set via the UU command.



Example: Make sure the UU512 command we sent earlier is still current. The command will return the UU value with six digits to the right of the decimal point. If the UU value exceeds six digits for the fractional value, the value will be rounded off to the sixth decimal place.

Enter: ?UU

Response: <CR><LF>uu512.000000<CR><LF>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?UU	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

If user units are turned off ?UU returns:

<CR><LF>UF<CR><LF>

Related commands: UU, UF

5.4.7. AXIS STATUS COMMANDS

The PCIX monitors the various inputs and conditions that can affect motor movement and system status. This information is frequently needed by host applications so that proper motion decisions can be made and appropriate actions taken. The following commands provide this status feedback to the host.

QA

QUERY AXIS STATUS



The QA command returns the status of the single addressed axis like the RA command except the limit and done flags will not be cleared. Refer to the RA command for details.



Example: Check the status of the X axis.

Enter: AX QA

Response: <LF><CR><CR>PNNH<LF><CR><CR>

Refer to the table "Character Meaning" in the RA command on [page 5-29](#).

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
QA	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: QI, RA, RI, EA

QI**QUERY INTERRUPT STATUS**

The QI command returns the same information as the QA command but for all axes at once. The 4 character fields for each axis are separated by commas and the string has one line feed and two carriage returns at each end.



Example: Check the status of a four axis board.

Enter: AA QI

Response: <LF><CR><CR>PNNN,MNNN,PDNN,MNLN<LF><CR><CR>

Refer to the table "Character Meaning" in the RA command on [page 5-29](#).

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
-	AX – AT	Not Valid	
QI	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: QA, RA, RI, EA

RA REQUEST AXIS STATUS

The RA command returns the state of the limit and home switches and the done and direction flags for the currently addressed axis. The limit flag in the hardware status register will be reset by the RA command, provided another axis is not in limit. The done flag register will also be reset by this command. The status is returned in the following format:

CHARACTER MEANING		
CHAR	SENT	DESCRIPTION
1	LF	Line feed
2	CR	Carriage return
3	CR	Carriage return
4	P	Moving in positive direction
	M	Moving in negative direction
5	D	Done (ID, II or IN command has been executed, set to N by this command or IC command)
	N	No ID executed yet
6	L	Axis in overtravel. Char 4 tells which direction. Set to N when limit switch is not active.
	N	Not in overtravel in this direction
7	H	Home switch active. Set to N when home switch is not active.
	N	Home switch not active
8	LF	Line feed
9	CR	Carriage return
10	CR	Carriage return



Example: The Y axis just encountered a limit, verify its status.

Enter: AY RA

Response: <LF><CR><CR>PNLN<LF><CR><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
RA	AX – AT	Immediate	
RA	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: QA, QI, RI, EA

RI REQUEST INTERRUPT STATUS

The RI command is a multiaxis mode command that returns the same status information for all axes as the RA command does in single axis mode. The 4 character fields for each axis are separated by commas and the string has one line feed and two carriage returns on each end. The done flag is reset by this command as it would be via the RA command.



Example: Check the status of a 4 axis board.

Enter: AA RI

Response: <LF><CR><CR>MDNN, MDNN, PNLN, PNNN<LF><CR><CR>

Refer to the table "Character Meaning" in the RA command on [page 5-29](#).

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
-	AX – AT	Not Valid	
RI	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: QA, QI, RA, EA

EA ENCODER STATUS

The EA command returns encoder status of the currently addressed axis or axes in the following format:

EA COMMAND RESPONSE DESCRIPTION		
CHAR	SENT	DESCRIPTION
1	LF	Line feed
2	CR	Carriage return
3	CR	Carriage return
4	E	Slip detection enabled
	D	Slip detection disabled
5	E	Position maintenance enabled
	D	Position maintenance disabled
6	S	Slip or stall detected (reset by execution of EA command)
	N	No slip or stall detected
7	P	Position Maintenance within deadband
	N	Position not within deadband
8	H	Axis is home
	N	Axis is not home
9	N	Unused/reserved
10	LF	Line feed
11	CR	Carriage return
12	CR	Carriage return



Example: Examine the status of the Y axis encoder.

Enter: AY EA

Response: <LF><CR><CR>EENPNN<LF><CR><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
EA	AX – AT	Immediate	
EA	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: QA, QI, RA, RI

5.4.8. MACROS

In applications that must perform frequent, repetitive tasks, macros can be used to minimize communication bandwidth consumption and speed up initial task execution. Macros are storage areas in the PCIX of which 5 are “temporary”; i.e. not saved at power-off, and 20 are “permanent”; i.e. stored in non-volatile flash RAM.

Macros can be edited, stored, read back to the host, and executed using the following commands.

MD

TEMPORARY MACRO DEFINE



MD is used to begin defining a temporary macro. A macro can contain up to 250 characters. Macros 0 through 4 are temporary and they will be erased when the controller is reset or power is turned off. Macros 5 through 24 are stored in non-volatile memory and will be preserved when the controller is reset or powered off. This command cannot be used to define a macro directly into numbers 5 through 24. They must be defined with this command and then moved into the nonvolatile space with the PT command.

Enter the macro number immediately after the MD command. The macro number must be between 0 and 4. Next enter the command string, which is made up of up to 250 ASCII characters. After entering the command string for the macro, enter a control Z to end the macro definition. The control Z may be ASCII value 26 or the string “^Z” (carat, shift 6 on a US keyboard, followed by a Z.)

Be careful not to exceed 250 ASCII characters or the size of the axis queue when working with macros.

RANGE: 0 <= MD <= 4



Example: Define macro 2 to set velocities to 20000 on all axes of a two axis board.

Enter: MD2 ;
AA VL20000,20000 ;
^Z

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
MD#;	AX – AT	Immediate	
MD#;	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: PT, MX, PM

PT**PRESERVE A TEMPORARY MACRO**

Use PT to save a temporary macro permanently by copying it to non-volatile memory. The temporary macro number, which is entered as an argument for this command, must be between 0 and 4. The non-volatile macro number, which is also entered as an argument for this command, must be between 5 and 24.

RANGE:

0 <= Parameter 1 <= 4

5 <= Parameter 2 <= 24



Example: Copy temporary macro 3 to non-volatile macro 19.

Enter: PT3 , 19 ;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
PT#,#;	AX – AT	Immediate	
PT#,#;	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: MD, MX, PM

MX**MACRO EXECUTE**

The MX command will execute the command string stored in the specified macro. The macro number that is entered as the argument of the command must be between 0 and 24.

RANGE: 0 <= MX <= 24



Example: Execute macro number 6.

Enter: MX6 ;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
MX#;	AX – AT	Immediate	
MX#;	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: MD, PT, PM

PM PRINT MACRO

The PM command will return the command string stored in the specified macro number as a command response. The macro number entered as the argument for this command must be between 0 and 24.

RANGE: 0 <= PM <= 24



Example: Print the command string contained in macro 19.

Enter: PM19 ;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
PM#;	AX – AT	Immediate	
PM#;	AA-AM	Immediate	
-	AA/CD	Not Valid	



NOTE: Macros are stored as ASCII character strings. If <CR><LF> characters are used as command terminators they will be sent back to the host computer by the PM command. If the application software stops reading a character string at the first <CR><LF> it will appear that the PM command did not return the macro contents. To avoid this issue just save macros without <CR><LF> delimiters. Use semi-colons instead.

Related commands: MD, PT, MX

5.5. I/O CONTROL COMMANDS

5.5.1. AUXILIARY CONTROL COMMANDS

Each axis of the PC1x has an associated auxiliary output line. Though this line can be used as a general purpose output, it also has a special purpose: Power-Automatic Mode. In power-automatic mode, the auxiliary line will invert at the beginning of every motion and return to normal at the end. The “normal” state of this line is user-controllable as is the amount of time to delay, allowing the motor to settle, before returning the line to normal at the end of a move. The following commands provide this control as well as feedback regarding the state and function of each auxiliary line.

PA

POWER AUTOMATIC



The PA command will turn on or off the auxiliary outputs at the beginning of each GO or GD command execution and complement the outputs after the move is executed. The auxiliary will be turned on; i.e. pulled high, upon the execution of the GO or GD and off at the end of that move if the parameter is zero or not specified in the single axis mode. If the parameter is non-zero, the sense is reversed; i.e. the auxiliary output is turned off (driven low) upon the execution of the GO or GD command and on at the end of the move.

The SE command can be used to apply a settling time at the end of each move before complementing the auxiliary bit. This is useful for systems that need to retain torque for some specific amount of time before allowing the motor drive to reduce current output.

This mode need only be set once and can be turned off by using the AN or AF command. Axes can be selectively affected in the AA mode. The values of the included parameters set the state of the auxiliary line during the move. The following queue requirements apply to each GO or GD command in the command stream in the AA and single axis modes. This mode is off by factory default. See the AP Command on [page 5-10](#) to preserve the PA settings as the Power up/Reset values.



Example:

Turn on the Y axis auxiliary output at the beginning of a move and turn the T axis output off at the beginning of a move, while in the AA command mode. (note the reversed logic; i.e. 0 = on, 1 = off. "on" pulls the signal line to ground. "off" lets it rise to 5 volts or its pull-up reference voltage.)

Enter:

```
AA PA, 0, , 1;
```

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
PAb;	AX – AT	1*	4*
PAb,b,b,b;	AA-AM	1*	4*
-	AA/CD	Not Valid	

* AX-AT/ AA-AM

- If an auxiliary output bit settle time has been specified add 3 to the queue requirement.

Related commands: ?PA, AN, AF, SE, ADH, ADL

?PA**REPORT POWER AUTOMATIC STATE**

The ?PA command reports whether the current axis has power automatic mode enabled. The PA command is used to set this value. This command will respond PA0 if power automatic mode is disabled and PA1 if it is enabled.



Example: Power automatic mode should be enabled for axis X. Use ?PA to find out whether this is the case.

Enter: AX ?PA

Response: <LF><CR>PA1<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
-	AX – AT	Immediate	
?PA	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: PA, ?SE

AN**AUXILIARY ON**

The AN command turns on the selected auxiliary output ports. That is, it allows the signal to be pulled high. This is the default mode for the auxiliary line at power up or reset. The AN command may be used to change power level on driver modules so equipped, trigger another board's input or as a user specified output.

A parameter must be supplied for the desired axes when used in the AA mode so that the other axes are not affected. No parameter is required in a single axis mode. Note this command will turn power automatic (PA) mode off.



Example: Turn on the Y axis auxiliary output in the single axis mode.

Enter: AY AN



Example: Turn on the X and Z axes auxiliary outputs when in the AA command mode. The Y axis is unchanged in this example.

Enter: AA AN1 , , 1 ;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
AN	AX – AT	1	1
ANb,b,b,b;	AA-AM	1	1
ANb,b,b,b;	AA/CD	2	2

Related commands: AF, PA, BH, BL, BS

AF**AUXILIARY OFF**

The AF command turns off the selected auxiliary outputs. That is, it causes the signal to be driven low. The AF command may be used to change power level on driver modules so equipped or as a user specified output. Note that this command will turn power automatic (PA) mode off.



Example: Turn off the Y axis auxiliary output in the single axis mode.

Enter: AY AF



Example: Turn off the X and Z axes auxiliary outputs when in the AA command mode. The Y axis is unchanged in this example.

Enter: AA AF1, , 1;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
AF	AX – AT	1	1
AFb,b,b,b;	AA-AM	1	1
AFb,b,b,b;	AA/CD	2	2

Related commands: AN, PA, BH, BL, BS

SE SETTLING TIME

The SE command allows specification of a settling time, in milliseconds, to be used before the auxiliary output is complemented when using PA mode. The parameter may be any value up to 1000 milliseconds. Specification of a parameter of zero disables SE mode.

The use of this command requires 3 queue slots with the execution of each GO or GD command. The factory default settling time is zero. See the AP Command on [page 5-10](#) to preserve the SE settings as the Power up/Reset values.

RANGE: 0 <= SE <= 1000



Example: Turn on the Z axis auxiliary output upon execution of a move and have it remain on for 500 milliseconds after the move is complete.

Enter: AZ PA SE500;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
SE#;	AX – AT	Immediate	
SE#,#,#,#;	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: ?SE, AN, AF, PA

?SE**REPORT SETTLING TIME**

The ?SE command reports the settling time setting (SE) used with power automatic mode (PA) for the current axis.



Example: Report the current settling time for axis X

Enter: AX ?SE

Response: <LF><CR>SE250<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?SE	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: SE, ?PA

ADH SET AUXILIARY DEFAULT TO HIGH



The ADH command sets the default powerup or reset state of the auxiliary line for the current axis to high. This change is stored in nonvolatile memory automatically and need not be stored via the AP command. Since this command writes to non-volatile memory it should be used only when necessary and not in repeatedly called functions.



Example: Set the powerup state of the Z axis auxiliary line to high

Enter: AZ ADH

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
ADH	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ADL, ?AD

ADL SET AUXILIARY DEFAULT TO LOW

The ADL command sets the default powerup or reset state of the auxiliary line for the current axis to low. This change is stored in nonvolatile memory automatically and need not be stored via the AP command. Since this command writes to non-volatile memory it should be used only when necessary and not in repeatedly called functions.



Example: Set the powerup state of the Y axis auxiliary line to low

Enter: AY ADL

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
ADL	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ADH, ?AD

?AD**REPORT DEFAULT AUXILIARY
BIT STATE**

This command will reply with the current powerup default state of the current axis' auxiliary bit as set by ADH or ADL. The response will be either ADH or ADL based on the current powerup state of the auxiliary bit.



Example: Report the powerup state of the Y axis auxiliary bit

Enter: AY ?AD

Response: <LF><CR>ADL<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?AD	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ADL, ADH

5.5.2. GENERAL PURPOSE I/O CONTROL

The PCIx has 8 configurable general purpose I/O bits. From the factory they are configured as 4 inputs and 4 outputs. The following commands can be used to set outputs high or low individually or as a group and to read inputs as a group.

BL**BIT LOW**

The BL command turns the selected general purpose output line on (i.e. logic low). The default states of all output bits at power-up are logic high (off). The BS command can be used to set all outputs to a known state at once.



Note: Output bits should not be used as triggers for applying power to any device unless master power is applied separately and after the PCIx is fully configured. The states of the outputs are unpredictable during power-up and reset and can toggle several times before settling at a high level.



Example: Turn on output bits 4 and 5 after a move. Note that this is only valid for output bits; input bits cannot be modified.

Enter: AX MA1000; GO BL4; BL5;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
BL#;	AX – AT	2	2
BL#;	AA-AM	3	3
BL#;	AA/CD	2	2

Related commands: AN, AF, BX, BL, BS

BH**BIT HIGH**

The BH command turns the selected general purpose output off (i.e. logic high). The state of general purpose outputs is off at power up or reset.



Note: Output bits should not be used as triggers for applying power to any device unless master power is applied separately and after the PCIx is fully configured. The states of the outputs are unpredictable during power-up and reset and can toggle several times before settling at a high level.



Example: Set general purpose bits 4 and 5 to high.

Enter: BH4 ; BH5 ;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
BH#;	AX – AT	2	2
BH#;	AA-AM	3	3
BH#;	AA/CD	2	2

Related commands: AN, AF, BX, BL, BS

BS**BIT SET**

Set all of the output bits to a known state at the same time. This command will affect all output bits, setting their states to the specified mask nearly simultaneously. The mask must be in ASCII hex format where the least significant bit (bit 0) is on the right. To set a line low, the corresponding bit in the hex mask must be a 1. A zero (0) in any bit position will set the corresponding bit high.



Example: Set output 0 high, 1 - 4 low, and 5 - 7 high, (1E = (hex) 00011110)

Enter: BS1E;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
BS#;	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: AN, AF, BH, BL, BX

BX BIT REQUEST IN HEX

The BX command returns the states of the general purpose input bits in a two digit hex format, surrounded by linefeed + carriage return pairs. The rightmost character represents the least-significant-nibble (4 bits) and, if the nibble is rewritten as bits, the rightmost bit is the least-significant bit.

An input set low will be represented as a binary 1 and a high as binary 0.



Example: User input lines 0 and 2 are currently low and lines 1 and 3 are high. Use the BX command to verify these states, (5 = 0101).

Enter: BX

Response: <LF><CR>05<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
BX	AX – AT	Immediate	
BX	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: BS, BH, BL

5.5.3. HOME CONTROL COMMANDS

System homing is an essential step in most systems. To accommodate a wide range of homing methods, the following commands provide the ability to set home inputs active high or low and to enable or disable encoder index signals as part of home detection.

HH
HOME HIGH


The HH command sets the sense of the home switch on the current axis to active high. The default “true” state of the home input is a TTL logic low. This command allows a TTL logic high to be treated as the “true” state for applications where this is more convenient. Once this command has been sent to the PCIx, active high homes can be made the power-up default by using the AP command; refer to the AP command for details.



Example: (see HL command [on page 5-51](#))

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
HH	AX – AT	1	1
HHb,b,b,b;	AA-AM	1	1
-	AA/CD	Not Valid	

Related commands: HL, HE, HS, HM, HR, KM, KR

HL**HOME LOW**

The HL command sets the sense of the home switch on the current axis to active low. This is the power-up and reset default “true” state unless HH has been saved as a user power-up default (See the AP command). This command sets the “true” state of a home input to a TTL logic low if the current “true” state is a TTL logic high as it would be had an HH command been entered prior to this command.



Example: The stage is moved through home with the home switch set for active low at low speed to meet the less than 2048 steps per second requirement of the home command.

Enter: AX VL2000; HL HM0;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
HL	AX – AT	1	1
HLb,b,b,b;	AA-AM	1	1
-	AA/CD	Not Valid	

Related commands: HH, HE, HS, HM, HR, KM, KR

HE HOME ENCODER

The HE command enables encoder index to be considered in addition to the home switch input when an HM or HR command is executed. This command does not execute the home motion which must be initiated with an HM, HR, KM, or KR command. Default home behavior may be reestablished via the HS command. In this mode, home is defined as the logical AND of the encoder index, the external home enable and the encoder quadrant where channel A is positive and channel B is negative. The external enable is always low true. The HH and HL commands are not valid in this mode. The home logic expressed in Boolean terms is:

$$home = phase_A * / phase_B * index * / home_switch$$



Example: Set up the Y axis so it will use the encoder signals to recognize the home position.

Enter: AY HE

or

AA HE, 1 ;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
HE	AX – AT	Immediate	
HEb,b,b,b;	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: HS, HH, HL, HM, HR, KM, KR

HS HOME SWITCH

The HS command disables HE mode and returns the PCIX to the default home behavior. Default behavior defines a home state to be active when the home switch input is either low in default or HL mode or high when in HH mode. This mode can also be used with encoders which contain internal home logic by connecting their output to the PCIX home input for the appropriate axis. The active level of this input may be controlled by the HH and HL commands.



Example: Set up the Y axis so it will ignore the encoder signals and only use the home input to recognize the home position.

Enter: AY HS

or

AM HS, 1 ;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
HS	AX – AT	Immediate	
HSb,b,b,b;	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: HE, HH, HL, HM, HR, KM, KR

5.5.4. LIMIT CONTROL

Limit conditions are treated as critical errors in the PCIx. When a limit is encountered, the axis involved will cease motion and flush any pending motion commands for that axis. However, since needs vary from application to application, the following commands will allow limit behavior customization to fit almost any system.

LF

LIMITS OFF



The LF command disables the limit switches for the addressed axis or axes. This allows the stage to move beyond the limit switches and should be used with caution. The PCIx will still recognize that a limit switch has been closed if the stage runs into one and will report this information via the query commands (See the QA command). However, the limit switch closure will have no effect on motion; i.e. the axis will not be forced to stop as a result.



NOTE: In systems not designed to handle motion beyond the limit switch points, this can potentially cause damage to the system and/or persons operating the system. This command should be used with extreme caution.



Example: Set up a board to ignore the Y axis limit switches.

Enter: `AY LF`

or

`AA LF, 1;`

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
LF	AX – AT	1	1
LFb,b,b,b;	AA-AM	1	1
-	AA/CD	Not Valid	

Related commands: LN, LH, LL, SL, SF

LN LIMITS ON

The LN command restores the operation of the limit switches for the addressed axis or axes. This is the default mode at power up or reset. With limit switches enabled, if the axis encounters a limit switch in the course of executing any motion, the axis will be instructed to stop and the host will be notified of the event. Limit conditions are treated as “critical errors” and should not be used as simple positioning inputs to the PCIX.



Example: Set up the Y and T axes to stop immediately when a limit switch is encountered.

Enter: AA LN, 1, , 1;

or

AY LN AT LN

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
LN	AX – AT	1	1
LNb,b,b,b;	AA-AM	1	1
-	AA/CD	Not Valid	

Related commands: LF, LH, LL, SL, SF

LH LIMITS HIGH

The LH command sets the senses of the limit switches on the current axis to active high. The default “true” states of the limits are TTL logic low. This command allows a TTL logic high to be treated as the “true” state for applications where this is more convenient. Through the execution of the AP command, limits can be made to default to active high on power-up or reset; see the AP command for details.



Example: Select the limit switch high true condition for the X axis.

Enter: AX LH



Example: Select a high true limit condition for the Z and T axes.

Enter: AA LH, , 1, 1;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
LH	AX – AT	1	1
LHb,b,b,b;	AA-AM	1	1
-	AA/CD	Not Valid	

Related commands: LN, ?LS, LF, LL, SL, SF, TL

LL LIMITS LOW

The LL command specifies that overtravel occurs when the limit input signal is low (active low). This is the factory default mode. See the AP Command on page 5-10 to preserve the LL settings as the Power up/Reset mode.



Example: Select the limit switch low true condition for the X axis.

Enter: AX LL



Example: Select a low true limit condition for the Y and T axes.

Enter: AA LL, 1, , 1;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
LL	AX – AT	1	1
LLb,b,b,b;	AA-AM	1	1
-	AA/CD	Not Valid	

Related commands: LN, ?LS, LF, LH, SL, SF, TL

?LS**REPORT LIMIT ACTIVE STATE**

The ?LS command reports the active state of the limits for the current axis. The LL and LH commands are used to set this value and are the possible responses of this command.



Example: Find out whether the Y axis limits are active high or active low.

Enter: AY ?LS

Response: <LF><CR>LL<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?LS	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: LH, LL

SL SOFT LIMITS ON

The SL command enables the PCIX to ramp an axis to a stop rather than abruptly killing the motion when a limit switch is encountered on that axis. The output queue is not flushed except for the current move. This mode is effective for point to point and JG moves only. Soft limits can be made the power-up default via the AP command.



Example: Set up a board to allow the X axis to ramp to a stop when a limit is encountered.

Enter: AX SL



Example: Set up a board I/O to allow the Y and T axes to ramp to a stop when a limit is encountered.

Enter: AA SL, 1, , 1;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
SL	AX – AT	5*	10*
SLb,b,b,b;	AA-AM	5	10
-	AA/CD	Not Valid	

* AX-AT

- If the controller has an encoder or servo axis add 1
- If PA mode is active add 1
- If an auxiliary output bit settle time has been specified add 3

Related commands: SF, ?SL, LN, LF, LH, LL, TL

SF**SOFT LIMITS OFF**

The SF command restores the default operation of the limit switches; i.e. causes the affected axis or axes to abruptly halt when a limit switch is encountered. If soft limits have been made the power-up default, the AP command must be used to store hard limit operation as the default.



Example: Set up a board to make the X axis stop immediately when a limit is encountered.

Enter: AX SF



Example: Set up a board to make the Y and T axes to stop immediately when a limit is encountered.

Enter: AA SF,1,,1;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
SF	AX – AT	1	1
SFb,b,b,b;	AA-AM	1	1
-	AA/CD	Not Valid	

Related commands: SL, ?SL, LN, LF, LH, LL, TL

?SL REPORT SOFT LIMIT STATUS

The ?SL command reports whether soft limits are currently enabled for the active axis. Possible responses are SL and SF, the same commands used to enable and disable soft limits.



Example: Find out whether soft limits are enabled for axis Z

Enter: AZ ?SL

Response: <LF><CR>SL<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?SL	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: SL, SF

TL SET SOFTWARE TRAVEL LIMITS

The TL command sets logical limits on the range of travel for an axis. Two parameters must be supplied; one for the upper travel limit and the other for the lower travel limit, both as absolute positions. If the axis reaches either of these logical limits, the PClx will flag a limit condition just as it would using the physical limit switch inputs. Move Relative (MR) type motion is subject to software travel limits because the PClx checks an internal absolute position register.

Set both parameters equal to disable software travel limits.

RANGE:

-33,554,000 <= Parameter 1 <= 33,554,000 (Upper Position Limit)

-33,554,000 <= Parameter 2 <= 33,554,000 (Lower Position Limit)



Example: Set logical position limits for the X axis of +/-1,000,000.

Enter: AX TL1000000,-1000000;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
TL#,#;	AX – AT	3	3
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ?TL, LL, LH, LN, LF, SL, SF

?TL**REPORT SOFTWARE
TRAVEL LIMITS**

The ?TL command reports the software travel limits for the current axis assigned via the TL command. The first value returned is the upper (or "positive") limit and the second value is the lower (or "negative") limit. These are not physical limits but rather positional limits that an axis should not exceed. If one of these limits is exceeded, the PCIX will set the current axis' limit flag and notify the host computer of the condition as though the axis encountered a hard limit.



Example: Find out what the software limits of the Y axis are currently set to.

Enter: ?TL

Response: <LF><CR>TL101000, -1000<LF><CR>



Example: Find out what the software limits of the T axis are currently set to.

Enter: AT ?TL

Response: <LF><CR> TL0, 0; <LF><CR> (software limits for axis T are currently disabled)

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?TL	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: TL

5.6. SERVO CONTROL COMMANDS

The following commands are valid only for servo axes and should never be executed while the specific axis is in motion.

5.6.1. SERVO VOLTAGE CONTROL COMMANDS

Different servo amplifiers have different requirements for their control inputs. Some simply behave differently despite similar input requirements. To enable the use of a wide range of amplifiers, the PCIx will accept the following commands for use in configuring servo outputs.

BI
BIPOLAR


The BI command sets the analog servo output of the current axis to bipolar. When bipolar is selected, a zero torque reference will result in a 0VDC output (+/- offset voltage). The analog output will range between +10VDC and -10VDC when bipolar is enabled. The BI command is valid only in the single axis mode and is the default mode at power up or reset.



Example: Set up servo axis X for bipolar operation.

Enter: AX BI

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
BI	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: UN, ?SO, SVI, SVN, DBI, DBN

UN UNIPOLAR

The UN command sets the analog torque outputs of servo axes to unipolar. The analog output will range between 0.0VDC and +10VDC when unipolar is enabled. At maximum positive velocity, the board outputs +10VDC. At Maximum negative velocity, the board output approaches 0.0VDC. To maintain position the board outputs 5VDC. This command is valid only in single axis mode.



Example: Set up servo axis X for unipolar operation.

Enter: AX UN

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
UN	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: BI, ?SO, SVI, SVN, DBI, DBN

?SO REPORT ANALOG OUTPUT MODE

The ?SO command reports whether the analog output type for the current servo axis is bipolar or unipolar. The possible responses are BI and UN, the same commands used to set one mode or the other.



Example: The Y axis should be setup with unipolar outputs. Use ?SO to make sure.

Enter: AY ?SO

Response: <LF><CR>UN<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?SO	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: UN, BI

SVI INVERT SERVO VOLTAGE

The SVI command inverts the voltage output for the current axis. After receiving this command, the PCIx will produce a negative voltage for positive motion and a positive voltage for negative motion. To cancel this command, issue an SVN command. To make inverted servo outputs the default at powerup or reset, use the AP command.



Example: The Y axis encoder is counting opposite the expected direction. Setup the Y axis to produce a negative voltage when moving positive instead of a positive voltage to correct the problem.

Enter: AY SVI

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
SVI	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: SVN, ?SV, UN, BI, DBI, DBN

SVN NORMALIZE SERVO VOLTAGE

The SVN command normalizes the voltage output for the current axis, negating the effects of the SVI command. After receiving this command, the PC1x will produce a positive voltage for positive motion and a negative voltage for negative motion, the default behavior. To make this the default behavior (if it has been changed via SVI/AP), use the AP command. (SVN is the factory default setting.)



Example: The Y axis encoder was rewired and now counts in the correct direction. Return the Y axis servo output to normal; i.e. output positive voltage for positive motion.

Enter: AY SVN

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
SVN	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: SVI, ?SV, UN, BI, DBI, DBN

?SV**REPORT SERVO VOLTAGE
INVERSION STATE**

The ?SV command reports the current logical direction for the current servo axis. The state is set with the SVI and SVN commands.



Example: Report whether servo voltage is positive for positive moves on axis X

Enter: AX ?SV

Response: <LF><CR>svn<LF><CR> (voltage is normal; i.e. positive for positive moves)

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?SV	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: SVI, SVN, ?DB

DZ DEFINE ZERO POSITION IN OPEN-LOOP MODE



The DZ command defines the offset coefficient needed to produce a zero voltage output or stationary position by the servo motor. This command is used in the open-loop mode with Hold Off (HF). This command can be used in combination with or in lieu of the balance adjustment on the servo amplifier. This is particularly useful with amplifiers that do not have a balance adjustment.

The factory default value is zero. Full-scale, the DZ command has a range of +/- 32,667 which corresponds directly to the 16-bit range of the DAC less a few counts as a buffer zone. Each increment/decrement of the DZ value will result in an approximate change in the output voltage of 0.0003 volts. See the AP Command [on page 5-10](#) to preserve the DZ settings as the Power up/Reset values.

This command affects the offset only when in open-loop mode. The KO command is used to set the offset when the loop is closed. Typically, the value found that is satisfactory with the DZ command will also be used with the KO command. Since this is not always the case, DZ and KO are available to set differing offsets.

RANGE: -32,667 <= DZ <= 32,667



Example: Define the offset coefficient to be 250 (~ +76mV) for the X axis.

Enter: AX DZ250;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
DZ#;	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ?DZ, KO, HN, HF

?DZ**REPORT DAC OPEN-LOOP OFFSET**

The ?DZ command reports the current setting of the DZ command in DZ command format.



Example: The closed-loop offset needs to be set the same as the open-loop offset. We've already set the open-loop offset but forgot what value we used. Send the ?DZ command to find out.

Enter: ?DZ

Response: <LF><CR>DZ28<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?DZ	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: DZ, ?KO

KO**OFFSET COEFFICIENT IN
CLOSED-LOOP MODE**

The KO command defines the offset coefficient to cause the motor to remain stationary and compensate for additional torque on the motor from loading. This command is used in the closed-loop mode, hold on (HN). The factory default value is zero. See the AP Command [on page 5-10](#) to preserve the KO settings as the Power up/Reset values.

The factory default value is zero. Full-scale, the KO command has a range of +/- 32,667 which corresponds directly to the 16-bit range of the DAC less a few counts as a buffer zone. Each increment/decrement of the KO value will result in an approximate change in the output voltage of 0.0003 volts. See the AP Command [on page 5-10](#) to preserve the KO settings as the Power up/Reset values.

This command affects the offset only when in closed-loop mode. The DZ command is used to set the offset when the loop is open. Typically, the value found satisfactory with the KO command will also be used with the DZ command. Since this is not always the case, KO and DZ are available to set differing offsets.

RANGE: -32,667 <= KO <= 32,667



Example: Define the offset coefficient to be -2000 (~ -610mV) on the Y axis.

Enter: `AY KO-2000;`

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
KO#;	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ?KO, DZ, HN, HF

?KO REPORT PID CLOSED- LOOP OFFSET



The ?KO command reports the closed-loop voltage offset (KO) setting for the current servo axis. See the ?DZ command for reporting open-loop offset.



Example: The open-loop offset is 218. Make sure the closed-loop offset is the same.

Enter: ?KO

Response: <LF><CR>KO218<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?KO	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: KO, ?DZ

DBI**INVERT DIRECTION BIT**

The DBI command inverts the logic of the direction control output of the addressed axis or axes. By default, the direction output of an axis is a TTL low when traveling in the positive direction and high when traveling negative. After using the DBI command, the direction bit will be high when traveling positive and low when traveling negative. This is useful for inverting the logical direction of a motor when the encoder counts opposite the motor direction. This command can be canceled using the DBN command. To make this the default at powerup or reset, use the AP command.



Example: Set the direction outputs for axes Z and T to output high when traveling positive and low when traveling negative. Leave X and Y as they are.

Enter: AZ DBI AT DBI

or

AA DBI , , 1, 1;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
DBI	AX – AT	1	1
DBIb,b,b,b;	AA-AM	1	1
-	AA/CD	Not Valid	

Related commands: DBN, ?DB, UN, BI, SVI, SVN

DBN**NORMALIZE DIRECTION BIT**

The DBN command normalizes the logic of the direction control output of the addressed axis or axes, returning their output logic to default; i.e. TTL low when traveling in the positive direction and high when traveling negative. This command negates the effect of the DBI command. To make this the default at powerup or reset when DBI has already been made the default, use the AP command.



Example: Set the direction outputs for axes Z and T to default output logic; i.e. output low when traveling positive and high when traveling negative. Leave X and Y as they are.

Enter: AZ DBN AT DBN

or

AA DBN, , 1, 1;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
DBN	AX – AT	1	1
DBNb,b,b,b;	AA-AM	1	1
-	AA/CD	Not Valid	

Related commands: DBI, ?DB, UN, BI, SVI, SVN

?DB**REPORT DIRECTION BIT LOGIC**

The DB command returns the command, either DBI or DBN, that would set the direction bit to its current logical operation. If the direction bit is low when moving positive, this command will return the DBN command. If the direction bit has been inverted, this command will return DBI.



Example: Report whether the direction bit for the T axis is low or high when making positive moves

Enter: AT ?DB

Response: <LF><CR>DBI<LF><CR> (The DBI result indicates the T axis direction bit is high for positive moves)

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?DB	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: DBI, DBN, ?SV

5.6.2. PID COMMANDS

The PC1x uses a PID filter for servo position maintenance. The following commands provide user-control over the filter parameters and feedback of the same. See section [2.6 Connect and Checkout the Servo System on page 2-7](#) for more information regarding the use of these commands for tuning your servo motors.

KA

ACCELERATION FEEDFORWARD



KA is the acceleration feedforward coefficient used in the PID filter calculations. Refer to Section 2 for more information regarding this parameter. The factory default value is zero. See the AP Command on [page 5-10](#) to preserve the KA settings as the Power up/Reset values.

RANGE: 0 <= KA <= 249.99



Example: Define KA to be 2 on the T axis.

Enter: AT KA2;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
KA#;	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ?KA, KV, KP, KI, KD, HN, HF

?KA**REPORT ACCELERATION
FEED-FORWARD**

The ?KA command reports the current setting of the acceleration feed-forward constant (KA) for the current servo axis.



Example: Find out what the current KA value is for servo axis Y

Enter: AY ?KA

Response: <LF><CR>KA10.50000<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?KA	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: KA, ?KV

KV VELOCITY FEEDFORWARD

KV is the velocity feedforward coefficient used in the PID filter calculations. See Section 2 for a more detailed explanation of this parameter. The factory default value is zero. [See the AP Command on page 5-10](#) to preserve the KV settings as the Power up/Reset values.

RANGE: $0 \leq KV \leq 249.99$



Example: Set KV to 35.3 on the Y axis.

Enter: `AY KV35.3;`

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
KV#;	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ?KV, KA, KP, KI, KD, HN, HF

?KV**REPORT VELOCITY FEEDFORWARD**

The ?KV command reports the current velocity feed-forward setting (KV) of the current servo axis.



Example: Make sure the velocity feed-forward setting of axis T is zero

Enter: AT ?KV

Response: <LF><CR>KV0.00000<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?KV	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: KV, ?KA

KP**PROPORTIONAL GAIN COEFFICIENT**

KP is the proportional gain coefficient used in the PID filter calculations. See Section 2 for detailed information regarding this parameter. The factory default value is 10.00. [See the AP Command on page 5-10](#) to preserve the KP settings as the Power up/Reset values.

RANGE: 0 <= KP <= 249.99



Example: Define KP to be 45.6 on the Z axis.

Enter: `AZ KP45.6;`

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
KP#;	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ?KP, KI, KD, HN, HF

?KP**REPORT PID PROPORTIONAL GAIN**

The ?KP command reports the current setting of the proportional gain constant (KP) in the PID of the current servo axis.



Example: Find out what the X axis proportional gain is set to.

Enter: AX ?KP

Response: <LF><CR>KP10.00000<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?KP	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: KP, ?KI, ?KD

KI INTEGRAL GAIN COEFFICIENT

KI is the integral gain coefficient used in the PID filter calculations. See [Section 2](#) for more information regarding this parameter. The factory default value is 0.04. [See the AP Command on page 5-10](#) to preserve the KI settings as the Power up/Reset values.

RANGE: 0 <= KI <= 249.99



Example: Define KI to be 3.42 on the X axis.

Enter: AX KI3.42;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
KI#;	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ?KI, KP, KD, HN, HF

?KI**REPORT PID INTEGRAL GAIN**

The ?KI command reports the current setting of the integral gain constant (KI) in the PID of the current servo axis.



Example: Report the setting of the KI command on the Z axis

Enter: AZ ?KI

Response: <LF><CR>KI0.01000<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?KP	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: KI, ?KP, ?KD

KD**DERIVATIVE GAIN COEFFICIENT**

KD is the derivative gain coefficient used in the PID filter calculations. See Section 2 for more information regarding this parameter. The factory default value is 20.00. See the AP Command [on page 5-10](#) to preserve the KD settings as the Power up/Reset values.

RANGE: 0 <= KD <= 249.99



Example: Set KD to 56 on the Z axis.

Enter: AZ KD56;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
KD#;	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ?KD, KP, KI, HN, HF

?KD**REPORT PID DERIVATIVE GAIN**

The ?KD command reports the current setting of the derivative gain constant (KD) in the PID of the current servo axis.



Example: Forgot to write down the KD setting which is working well. Report the setting so it can be recorded.

Enter: ?KD

Response: <LF><CR>KD5.12500<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?KD	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: KD, ?KP, ?KI

HN**HOLD ON**

The HN command enables position correction after a move and activates the HV, HG and HD commands for stepper axes with encoders. For servo axes, the HN command closes the loop, enabling the PID. This mode will be canceled (as though via an HF command) if an LP, HM, HR, SA, ST or KL command is entered or if a limit is encountered.



Example:

The following commands could be used to set up the position correction mode on a stepper axis. This sequence sets up a move velocity of 100,000 steps per second and an acceleration of 500,000 steps per second per second. The position correction velocity is set for 50,000 steps per second, a deadband of 10 steps and correction gain of 2,000. The correction is then enabled. A 200,000 step move is performed, then that position is maintained within the 10 step deadband until commanded to a new position.

Enter:

```
AX VL100000; AC500000;
HV50000; HD10; HG2000; HN
MR200000; GO
```

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
HN	AX – AT	1*	3*
HN	AA-AM	1*	3*
-	AA/CD	Not Valid	

* AX-AT/AA-AM

- If this is a servo axis add 2

Related commands: HF, ?PM

HF**HOLD OFF**

The HF command disables position hold, stall detection and tracking modes. This is the default mode at power up or reset. If the current axis is a stepper with an encoder, this command will turn off position maintenance, stall detection and encoder tracking mode. If the current axis is a servo, this command will open the loop and turn off the PID. If the current mode is multiaxis, all axes will go into open-loop mode.



Example: Turn off encoder hold mode on the X axis.

Enter: AX HF

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
HF	AX – AT	2	2
HF	AA-AM	2	2
-	AA/CD	Not Valid	

Related commands: HN, ?PM

?PM**REPORT PID STATE**

The ?PM command reports whether the PID for the current servo axis is enabled. The HN and HF commands are used to enable and disable the PID and are the possible responses from this command.



Example: A limit switch was hit by servo axis Y. See if the PID is still enabled for that axis.

Enter: AY ?PM

Response: <LF><CR>HF<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?PM	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: HN, HF

5.7. STEP ENCODER CONTROL COMMANDS

5.7.1. STEP ENCODER CONTROL COMMANDS

Stepper systems, like servo systems, use encoder for position feedback. However, stepper systems do not use PID filters due to operating constraints in the stepper motors themselves. Instead, the PCIX uses the following commands to perform position maintenance for stepper axes.

It is important to note that stepper motor position cannot be maintained over the course of a move but rather at the end of the move. Once the axis has initially stopped, the axis will begin moving again to correct for any error encountered during the course of the full move. This process will continue until the encoder position is within the deadband of the motor's target position.

ER

ENCODER RATIO



The ER command allows specification of encoder:motor ratio for position maintenance mode. This command is *not* designed for use with servo motors. ER takes two arguments: encoder counts and motor counts. Both parameters *must* be integers unless user units (UU) are enabled. The ratio need not be per full revolution; reduce the fraction as far as possible and use those values.

If slip detection is enabled, this ratio must be 1:1. Any other ratio will disable slip detection. Slip detection mode is merely a notification mode where the host is notified whenever the motor and encoder positions vary by more than some specified amount. Position maintenance may use any ratio.

The factory default ratio is 1:1. [See the AP Command on page 5-10](#) to preserve the ER settings as the Power up/Reset values.

Parameter 1 = Encoder Counts
Parameter 2 = Motor Counts



Example: You have an encoder connected to a stepper motor through a series of gears. When the motor steps 25,000 times, the encoder produces 10,000 counts. Set up an encoder ratio so hold mode will work correctly.

Enter: `ER10000,25000;`

or

`ER2,5;`

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
ER#,#;	AX – AT	1	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ?ER

?ER REPORT MOTOR:ENCODER RATIO

The ?ER command reports the motor-to-encoder ratio as set with the ER command.



Example: Find out what the last ER command sent was.

Enter: ?ER

Response: <LF><CR>er2.000000<LF><CR>
(The encoder produces 1 count for every 2 steps of the motor.)

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?ER	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ER

HD**HOLD DEADBAND**

The HD command specifies deadband counts for position maintenance mode. If the encoder count is within this distance of target, it is considered in position and no further correction will be made. This parameter interacts with the HG and HV commands; i.e. a larger deadband will allow a larger gain parameter in many applications. This command is designed to work with stepper motor applications using encoders and is not designed for use with servo motors. The factory default value is zero. [See the AP Command on page 5-10](#) to preserve the HD settings as the Power up/Reset values.

RANGE: 0 <= HD <= 60,000



Example: (see HN command [page 5-87](#))

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
HD#;	AX – AT	2	2
HD#,#,#,#;	AA-AM	2	2
-	AA/CD	Not Valid	

Related commands: ?HD, HV, HG, HN, HF

?HD**REPORT POSITION
MAINTENANCE DEADBAND**

The ?HD command reports the current setting of the HD command. This command will only work with stepper axes with encoders.



Example: Find out what HD was last set to.

Enter: ?HD

Response: <LF><CR>HD5<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?HD	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: HD, ?HG, ?HV

HV HOLD VELOCITY

The HV command specifies the maximum velocity to be used when correcting position error. The factory default setting is zero; some value must be set for position correction to occur at all. See the AP Command on [page 5-10](#) to preserve the HV settings as the Power up/Reset values. This command is not designed for use with servo motors.

Hold gain (HG) will be used to scale the HV value based on the total error that must be corrected. In most cases the HV value will never be reached unless the position error is very wide or the HG value is set very high.

RANGE: 0 <= HV <= 1,044,000



Example: (see HN command [page 5-87](#))

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
HV#;	AX – AT	2	2
HV#,#,#,#;	AA-AM	2	2
-	AA/CD	Not Valid	

Related commands: ?HV, HG, HD, HN, HF

?HV**REPORT POSITION
MAINTENANCE VELOCITY**

The ?HV command reports the current setting of the HV command for the current axis. This command works only with stepper + encoder axes.



Example: Check the peak correction velocity for the T axis

Enter: AT ?HV

Response: <LF><CR>HV20000<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?HV	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: HV, ?HG, ?HD

HG**HOLD GAIN**

The HG command allows the user to specify position hold gain parameter. This gain parameter is multiplied by the position error in determining the velocity during correction. The velocity used will not exceed the value set with the hold velocity (HV) command. This command is designed to work with stepper motor applications using encoders and is *not* designed for use with servo motors. The parameter should be set experimentally by increasing it until the system is unstable then reducing it slightly below the threshold of stability. The factory default value is 1. See the [AP Command on page 5-10](#) to preserve the HG settings as the Power up/Reset values.

RANGE: 1 <= HG <= 32,000



Example: (see HN command [page 5-87](#))

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
HG#;	AX – AT	2	2
HG#,#,#,;	AA-AM	2	2
-	AA/CD	Not Valid	

Related commands: ?HG, HD, HV, HN, HF

?HG**REPORT POSITION
MAINTENANCE GAIN**

The ?HG command reports the current setting of the HG command for the current axis. This command works only with stepper + encoder axes.



Example: Position corrections seem slow. Check the setting of HG to be sure it is correct.

Enter: ?HG

Response: <LF><CR>HG100<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?HG	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: HG, ?HV, ?HD

5.7.2. STEP ENCODER SLIP COMMANDS

In applications that require notification when a stepper motor slips beyond a given tolerance, the following commands will be of assistance. These commands do nothing to maintain position. Instead, they tell the PCIx to react to a slip condition by notifying the host or ceasing motion.

ES ENCODER SLIP TOLERANCE



The ES command parameter specifies tolerance before slip or stall is flagged in the status register and in the RL command response. The encoder may get off target by as much as this value before the PCIx will consider the axis slipped. This mode must be turned on with an IS command and off with an HF command. The factory default value is 1. This command is intended to be used with stepper motors and *not* servo motors. [See the AP Command on page 5-10](#) to preserve the ES settings as the Power up/Reset values.

RANGE: 1 <= ES <= 65,535



Example: Your application can tolerate being up to 5 steps from the desired position before the controlling program should be notified of a slip condition.

Enter: ES5; IS

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
ES#;	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ?ES, IS, RL, TN, TF

?ES**REPORT ENCODER
SLIP TOLERANCE**

The ?ES command reports the current setting of the ES command.



Example: Report the current deadband for encoder slip detection tolerance

Enter: ?ES

Response: <LF><CR>ES15<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?ES	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ES

IS INTERRUPT ON SLIP

The IS command enables the PCIx to interrupt the host on slip or stall detection if the appropriate bit has been set in the interrupt control register. Hold and slip detection are disabled if an LP, HM, HR, SA, ST or KL command is entered or if a limit is encountered. If a slip occurs, slip detection must be re-enabled. The factory default value is 1. This command is intended to be used with stepper motors and not servo motors.



Example: [\(see ES command on page 5-99\)](#)

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
IS	AX – AT	1	1
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ES, RL, TN, TF

RL REPORT SLIP STATUS

The RL command returns the slip detection status of all axes. An S is returned if slip has occurred for that axis, or else an N is returned. The results are bounded by a linefeed + carriage return pairs as in other status commands. The number of characters returned corresponds to the number of axes available on the board. This command is intended to be used with stepper motors with encoders and not with servo motors.



Example: On a four axis board, see if any axis has slipped.

Enter: RL

Response: <LF><CR>NNSN<LF><CR> (The Z axis has slipped.)

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
RL	AX – AT	Immediate	
RL	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: ES, IS, TN, TF

TN TURN ON SLIP KILL MODE

The TN command enables slip kill mode. In this mode, if the motor slips beyond the deadband set by the ES command, the PClx will kill motion on the axis that slipped as though a KL command had been issued to the axis. This mode can be disabled (default) with the TF command.



Example: X axis is sent on a move. Its encoder cable was not connected to the controller (oops!). The controller issues a KL (Kill) command to the X axis after receiving the slip interrupt. The slip interrupt is generated once the difference between the motor position counts and encoder counts exceed 20.

Enter: AX ES20; TN; IS; LP0;
 MA30; GO

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
TN	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: TF, ES, IS, RL

TF TURN OFF SLIP KILL MODE

The TF command disables slip kill mode (enabled with TN.)



Example: Slip kill mode is enabled but a move needs to be performed where slip is likely and not important for this move. Disable slip kill mode.

Enter: TF

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
TF	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: TN, ES, IS, RL

5.7.3. ENCODER SLAVE MODES

Encoder tracking modes connect a motor to an axis at a given ratio. For each turn of the encoder, the motor will move proportionately.

ET

ENCODER TRACKING



The ET command turns on the encoder tracking mode. The axis will track its encoder input, thus allowing one axis to follow the activity of another or a thumbwheel for manual positioning or the movement of another device that produces a signal compatible to the encoder inputs. No acceleration or deceleration ramps are generated. The axis will duplicate the encoder input. The ER command allows the user to scale the motor's movements relative to the encoder. This command is intended to be used with stepper motors with encoders and not with servo motors.



Example: Set up the X axis so it will follow its encoder input.

Enter: AX ET

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
ET	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ER, HF

Location Initialization Commands

These commands allow the initialization of the physical stage with the controller.

5.7.4. HOMING COMMANDS

[Section 5.5.3 Home Control Commands](#) details the commands available for customizing homing operations. The commands below initiate the physical homing process.

HM

HOME



The HM command will cause the current axis or specified axes to move in the positive direction at the predefined velocity until the home input line becomes true for each axis. The position counters will be initialized to the positions supplied as parameters.

The velocity should be less than 2048 counts per second to maintain accuracy of the home position loaded. A velocity of 2048 or less will provide a homing accuracy of +/-0 counts. A velocity of 2049 to 4096 will be accurate to +/-1 count. Every multiple of 2048 adds +/-1 count to the error range.

Each axis will, when home is detected, reset its position counter to the parameter specified in the HM command. Once the counter is reset, the axis will ramp to a stop at the rate specified previously via the AC command. This will result in the axes stopping beyond the home switch so care should be taken to ensure adequate stopping distance is available beyond the switches. The axes can be easily returned to the precise home switch position by using an MA command after the HM command.

If no parameter is specified in single axis modes, the HM command will use zero as a default value. Parameters must be specified in multi-axis modes to inform the PCIX which axes are to be homed.



Example:

Find the physical home position of the X axis of the stage. The motor runs until the home switch input is activated and then initializes the position counter to the parameter supplied. Since the motor decelerates to a stop after reaching home, it is necessary to do an MA to the same position as specified in the home command if it is desired to physically position the device at home. The following commands will find home, initialize it to 1000 counts, then return to home. In many cases it will not be necessary to return home, only find the position and synchronize the controller to it.

Enter: `AX VL1000; HM1000; MA1000; GO`

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
HM#;	AX – AT	4*	11*
HM#,#,#,#;	AA-AM	4*	11*
-	AA/CD	Not Valid	

* AX-AT

- If the controller has an encoder or servo axis add 2
- If PA mode is active add 2
- If an auxiliary output bit settle time has been specified add 3

* AA-AM

- If the controller has an encoder or servo axis add 2
- If PA mode is active add 2
- If an auxiliary output bit settle time has been specified add 3

Related commands: HR, HE, HS, HH, HL, KM, KR, LP, LO

HR**HOME REVERSE**

The HR command will cause the current axis to step in the negative direction at the predefined velocity, until the home input line goes true. It behaves exactly like the HM command, except it travels in the reverse direction.



Example:

In a long stage it may be awkward to travel the full distance to home at less than 2048 pulses per second. The following will get close to home at higher speed, then refine the position at lower speed in the reverse direction.

Enter: AX VL100000; HH HM; VL1000; HL HR;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
HR#;	AX – AT	4*	11*
HR#,#,#,#;	AA-AM	4*	11*
-	AA/CD	Not Valid	

* AX-AT/AA-AM

- If the controller has an encoder or servo axis add 2
- If PA mode is active add 2
- If an auxiliary output bit settle time has been specified add 3

Related commands: HM, HE, HS, HH, HL, KM, KR, LP, LO

KM**HOME AND KILL**

The KM command will move the current single axis in the positive direction until home is detected and then kill motion immediately; i.e. without using a deceleration ramp. The position counter will not be reset or cleared. Due to motor and/or payload inertia, the motor may not stop immediately but slip some distance instead. This will result in inaccurate position counters. This command's primary purpose is to move an axis out of the way quickly or to get the axis near home rapidly to speed up the homing process.



Example: Move the Y axis in the positive direction to the home sensor and stop movement as quickly as possible.

Enter: AY KM

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
KM	AX – AT	2*	7*
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

* AX-AT

- If PA mode is active add 2
- If an auxiliary output bit settle time has been specified add 3

Related commands: HM, HR, HE, HS, HH, HL, KR, LP, LO

KR HOME REVERSE AND KILL

The KR command will find home in reverse and stop generating pulses immediately; i.e. no deceleration ramp will be generated. This command is identical to the KM command except that the direction of motion is reversed.



Example: Move the Y axis in a negative direction to the home sensor and stop movement as quickly as possible.

Enter: AY KR

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
KR#;	AX – AT	2*	7*
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

* AX-AT

- If PA mode is active add 2
- If an auxiliary output bit settle time has been specified add 3

Related commands: HM, HR, HE, HS, HH, HL, KM, LP, LO

5.7.5. POSITION COUNTERS

Applications frequently need to know the actual positions of motors and encoders as opposed to the assumed positions the applications keep track of. The following commands are available for retrieving that information as well as forcibly setting those positions. This can be useful for setting “floating zero” positions.

LP

LOAD POSITION



The LP command will immediately load the number supplied as a parameter in the absolute position registers of the axis. In models with the encoder option, the parameter will be loaded into the encoder position register and the parameter times the encoder ratio will be loaded into the position counter. If no parameter is supplied, the value of zero is used. This command turns off the position hold and interrupt on slip modes when used in a PCIx with the encoder option.

The LO command can be used after this command to set the motor position independently of the encoder position.

RANGE: -33,554,431 <= LP <= 33,554,431



Example: The following would load the X axis position register with 1000, and the Z axis position register with 2000.

Enter: `AA LP1000,,2000;`



Example: The following would load the Y axis position register with 20,000 and the encoder position register with 30,000 counts, in encoder models.

Enter: `AY ER3,2; LP30000;`

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
LP#;	AX – AT	2*	4*
LP#,#,##;	AA-AM	2*	4*
-	AA/CD	Not Valid	

* AX-AT/AA-AM

- If the controller has an encoder or servo axis add 2

Related commands: LO, RM, RP, RE, RU

LO LOAD MOTOR POSITION

The LO command sets the motor position independently of the encoder position unlike LP which sets both to the same supplied value. The LP command will override the LO command and reset the motor position. If the LP command is used and a different motor position value than the encoder position is desired, the LO command must be reentered. Any valid position within the allowable range may be used.

RANGE: -33,554,431 <= LO <= 33,554,431



Example: Set the motor position to 50,000 and the encoder position to 100,000 on the T axis

Enter: AT LP100000; LO50000;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
LO#;	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: LP, RM, RP, RE, RU

RM**REMAINDER**

The RM command divides the position counter by the parameter supplied and replaces the position counter with the resulting remainder. The parameter must be greater than zero and less than 65,000. This command is used in applications where the controller is managing the motion of a continuously rotating object. It allows the position counter to keep track of the absolute position without regard to the number of revolutions it may have rotated. This command has no effect on the encoder position register on boards with the encoder feedback option.

RANGE: $0 < RM < 65,000$



Example: The current position of a rotating stage with a full-revolution count of 6000 is needed. Since this stage has been rotated several times without regard for the position, the position counter has reached 163,279. Send an RM6000 command to find out what the real position of the axis is.

Enter: (Current position is 163,279)
 RM6000;
 (Current position is now 1,279)

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
RM#;	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: LP, LO, RP, RE, RU

RP**REQUEST POSITION**

The RP command returns the current position of the currently addressed axis in single axis mode or all positions separated by commas in AA or AM mode. The position will be returned to the host in ASCII format. This command is not queued; i.e. the current position will be returned immediately even if the axis is in motion. The response is surrounded by linefeed + carriage return pairs.



Example: The current position on the Y axis is 12345. Use the RP command to verify the position.

Enter: AY RP

Response: <LF><CR>12345<LF><CR>



Example: Verify the positions of all 4 axes.

Enter: AA RP

Response: <LF><CR>100,200,300,400<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
RP	AX – AT	Immediate	
RP	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: RE, PP, PE

PP REPORT MOTOR POSITIONS

The PP command reports the motor positions of all axes in ASCII format. The response will be delimited by linefeed + carriage return pairs. All four axes will be reported even in single axis mode. (This is the same as AA RP)



Example: Report the motor positions of a four axes controller.

Enter: PP

Response: <LF><CR>0,0,0,125<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
PP	AX – AT	Immediate	
PP	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: RP, RE, PE

RU REPORT POSITION IN USER UNITS

The RU command returns the current position in user units (see UU command on [page 5-24](#)). The format of the response is a floating point number with five characters to the right of the decimal point. This response is surrounded by line feed and carriage return pairs.



Example: One revolution of a motor is 2000 steps. Define user units so moves can be referenced in revolutions. Move the Z axis 3 1/2 revolutions. Use RU to display the position when the move is complete.

Enter: AZ UU2000; LP0;
 MR3.5; GO
 (Wait until move is complete.)
 RU

Response: <LF><CR>3.50000<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
RU	AX – AT	Immediate	
RU	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: RP, RE, UU

RE**REPORT ENCODER POSITION**

The RE command returns the current encoder position of the currently addressed axis or axes in encoder counts. The ASCII string response is surrounded by linefeed + carriage return pairs.



Example: Examine the current encoder position of the Y axis.

Enter: AY RE

Response: <LF><CR>12345<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
RE	AX – AT	Immediate	
RE	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: PE, RP, PP

PE REPORT ENCODER POSITIONS

The PE command reports the encoder positions of all encoder and/or servo axes. The response is returned delimited by linefeed + carriage return pairs and is in ASCII format. All encoder positions will be reported even in single axis mode. (This is the same as AA RE.)



Example: Report the encoder positions of a four axes servo board.

Enter: PE

Response: <LF><CR>0,50,156,0<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
PE	AX – AT	Immediate	
PE	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: RE, RP, PP

5.8. PROFILE CONTROL COMMANDS

5.8.1. VELOCITY COMMANDS

Part of configuring any system involves defining velocity limits. The commands below provide control over these limits and allow feeding them back to the host.

VL

VELOCITY



The VL command sets the maximum velocity register of one or more axes to the operands which follow the command. The operands must be greater than zero and less than or equal to 1,044,000 steps per second, where the factory default is 200,000 steps per second. [See the AP Command on page 5-10](#) to preserve the VL settings as the power-up/reset values.

RANGE: 1 <= VL <= 1,044,000



Example: In single axis mode, set the X axis velocity to 10,000 counts per second.

Enter: AX VL10000;



Example: In the AA mode, set the peak velocities of the X and T axes to 5,000 and 50,000 respectively. Leave the other axes with their previous values.

Enter: AA VL5000,,,50000;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
VL#;	AX – AT	2*	26*
VL#,#,#,;	AA-AM	2*	26*
-	AA/CD	Not Valid	

* AX-AT/AA-AM

- PN0 & PR0 ramp selections add 2
- All other PN & PR ramp selections add 9
- CN & SC ramp selections add 9
- SR (Custom) ramp selections add (number of steps in ramp - 1)

Related commands: ?VL, VB, AC, DC

?VL **REPORT PEAK VELOCITY SETTING**

The ?VL command returns the peak velocity setting for the current axis as set by the VL command.



Example: Make sure our "AXVL50000;" command worked.

Enter: ?VL

Response: <LF><CR>v150000<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?VL	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: VL, ?VB

RV**REQUEST VELOCITY**

The RV command will return the current velocity at which the axis is moving. This may differ from the programmed maximum velocity if the axis is ramping up to speed or stopping. The response is surrounded by line feed and carriage return pairs. If the JF command is executing, the command only reports the integer part of the velocity.



Example: Jog the Y axis at 12345 steps per second.
Display the current velocity.

Enter: AY JG12345;
RV

Response: <LF><CR>12345<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
RV	AX – AT	Immediate	
RV	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: VL, ?VL, RC

VB

VELOCITY BASE



The VB command allows the acceleration ramp to start off at a specified velocity. This allows faster acceleration and the ability to pass through resonance quickly in some applications. The velocity jumps instantly to the specified velocity, then ramps as usual. The deceleration is the same in reverse. This mode is active only for linear ramps; it is ignored for cosine and parabolic ramps but not flagged as a command error. The parameter must be greater than zero and less than the programmed velocity, where the factory default is zero steps per second. This command is not valid with the JG command nor will it work in conjunction with the DC command. See the AP Command on [page 5-10](#) to preserve the VB settings as the power-up/reset values.

If the VL command is used after the VB command and the velocity value set with VL is less than the previously set VB value, the velocity base will be reset to the velocity value minus one. This will result in a one-step acceleration ramp and must be taken into consideration in applications making use of the VB command.

RANGE: $0 \leq VB < VL \text{ value}$



Example: In the single axis mode, set the Y axis velocity base to 200.

Enter: AY VB200;



Example: In the AA mode, set the X and Y axes velocity bases to 200.

Enter: AA VB200,200;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
VB#;	AX – AT	2	2
VB#,#,##;	AA-AM	2	2
-	AA/CD	Not Valid	

Related commands: ?VB, VL, AC, DC

?VB**REPORT BASE
VELOCITY SETTING**

The ?VB command returns the base (starting) velocity setting for the current axis as set by the VB command.



Example: The acceleration ramp should start at 0pps. Make sure we didn't leave it at some other value.

Enter: ?VB

Response: <LF><CR>vb1500<LF><CR> (Oops! We forgot to set it back to zero)

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?VB	AX – AT	Immediate	
	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: VB, ?VL

5.8.2. ACCELERATION COMMANDS

Along with velocity limits, acceleration limits are also critical to most systems. The following commands allow customization of these parameters.

AC

ACCELERATION



The AC command sets the acceleration/deceleration register to the operand which follows the command. The parameter must be greater than zero (zero is not valid) and less than 8,000,000, where 2,000,000 is the factory default, and the unit is in steps per second per second. All the following move commands for the axis being programmed will accelerate and decelerate at this rate until another AC command is entered. [See the AP Command on page 5-10](#) to preserve the AC settings as the power-up/reset values.

RANGE: 1 <= AC <= 8000000



Example: In the single axis mode, set the Y axis acceleration to 200,000 counts per second per second.

Enter: AY AC200000;



Example: In the AA mode, set the acceleration of the X axis to 200,000 and the Z axis to 50,000 and leave the other axes with their previous values.

Enter: AA AC200000,,50000;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
AC#;	AX – AT	6*	30*
AC#,#,#,##;	AA-AM	6*	30*
-	AA/CD	Not Valid	

* AX-AT/AA - AM

- PN0 & PR0 ramp selections add 2
- All other PN & PR ramp selections add 9
- CN & SC ramp selections add 9
- SR (Custom) ramp selections add (number of steps in ramp - 1)

Related commands: ?AC, RC, DC, VL, VB

?AC**REPORT AC COMMAND**

This command will reply with the current acceleration value for the current axis in an AC command format (see below).



Example: Report the current AC value for this axis.

Enter: ?AC

Response: <LF><CR>AC200000<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?AC	AX – AT	Immediate	
	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: AC

RC**REQUEST ACCELERATION**

The RC command will return the current acceleration rate of the current axis. This may differ from the programmed acceleration if a cosine (CN) or parabolic (PN) ramp is being generated. When the stage is stopped, the parameter returned will be zero (0). When the stage is running at programmed speed; i.e. not accelerating, the parameter returned will be zero (0). While a contour is executing, the value computed to generate the appropriate lead in will be returned. The response to the RC command is surrounded by linefeed + carriage return pairs.



Example: Display current acceleration values for all axes on a four axis board.

Enter: AA RC

Response: <LF><CR>2000000,2000000,2000000,2000000<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
RC	AX – AT	Immediate	
RC	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: AC, RV, ?AC

DC**DECELERATION**

The DC command sets a deceleration rate overriding the AC parameter when the GU command is used to initiate a move. Only the GU command will use the DC value. The deceleration rate defaults to 200,000 and will take on whatever value is entered via the AC command. Therefore, the DC command must be reentered after using AC if a different deceleration rate is desired.

RANGE: 1 <= DC <= 8000000



Example: Send the Y axis on a 100,000 count move that accelerates at 100,000 counts per second per second up to 50,000 counts per second and decelerates at 20,000 counts per second per second.

Enter: AY AC100000; DC20000; VL50000; MR100000; GU

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
DC#;	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: AC, RC, VL, VB, GU

5.8.3. PROFILE COMMANDS

Often, the default linear acceleration profile is not optimum for a given system. To meet the needs of those systems, the PClx has a number of commands that allow partial or even complete customization of the profile. The commands below allow the use of parabolic, cosine, and even custom ramps. [See section 5.8.4 Custom Profile Commands](#) for commands to define custom ramps.

PF
LINEAR ON

The PF command restores all axes to linear acceleration and deceleration ramps. This command should not be given while an axis is in motion or the results may not be predictable. This command affects all axes even if issued in the single axis mode. PF is the factory default setting. [See the AP Command on page 5-10](#) to restore the PF setting as the Power up/Reset mode.



Example: Turn off cosine or parabolic ramps, returning to linear.

Enter: PF

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
PF	AX – AT	11	11
PF	AA-AM	11	11
-	AA/CD	Not Valid	

Related commands: ?RT, LA, PN, PR, CN, SC, SR

LA LINEAR RAMP PER AXIS

The LA command specifies that the linear acceleration ramp is to be used by the selected axes. This is the factory default for all axes. [See the AP Command on page 5-10](#) to preserve the LA settings as the Power up/Reset values. This command is similar to the PF command but can be used to switch a single axis rather than all axes at once.



Example: Select a linear ramp for the X axis.

Enter: AX LA



Example: Select the linear ramp for the Y and T axes.

Enter: AA LA,1,,1;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
LA	AX – AT	11	11
LAb,b,b,b;	AA-AM	11	11
-	AA/CD	Not Valid	

Related commands: ?RT, PF, PN, PR, CN, SC, SR

PN

PARABOLIC ON



The PN command sets all axes to truncated parabolic ramps. This acceleration profile starts at 100% of the programmed acceleration and decreases in steps of 10% of the initial acceleration down to as low as 10%. The parameter supplied selects the number of steps. It must be in the range of 3 to 10 corresponding to 70% and 10% acceleration at the peak respectively. A parameter out of this range or no parameter supplied defaults to 70% or 3 steps. Note that the parameter is the number of steps, not the acceleration values. The larger number is a lower acceleration at the peak. This command should not be given while an axis is in motion or the results may not be predictable. This command affects all axes even if issued in the single axis mode. The PF command is used to return to the default linear motion profiles. [See the AP Command on page 5-10](#) to preserve the PN setting as the Power up/Reset ramp.

RANGE: 3 <= PN <= 10



Example: Set the board to be in the smoothest parabolic acceleration ramp.

Enter: PN10 ;

QUEUE REQUIREMENTS		
FORMAT	MODE	REQUIREMENT
PN0;	AX – AT	15
PN≠0;	AX – AT	29
PN0,0,0,0;	AA-AM	15
PN≠0,0,0,0;	AA-AM	29
-	AA/CD	Not Valid

Related commands: ?RT, PR, PF, LA, CN, SC, SR

PR PARABOLIC RAMP PER AXIS

The PR command defines parabolic acceleration/deceleration ramps for use with one or more axes. This command is similar to the PN command except that only the specified axes are affected. The AP command can be used to store the settings of PR as the power-up/reset defaults.

RANGE: 3 <= PR <= 10



Example: Select a 10 step parabolic ramp for the T axis.

Enter: AT PR10;



Example: Select a 10 step parabolic ramp for the Y axis and a 8 step parabolic ramp for the T axis.

Enter: AA PR,10,,8;

QUEUE REQUIREMENTS		
FORMAT	MODE	REQUIREMENT
PR0;	AX – AT	15
PR≠0;	AX – AT	29
PR0,0,0,0;	AA-AM	15
PR≠0,0,0,0;	AA-AM	29
-	AA/CD	Not Valid

Related commands: ?RT, PN, PF, LA, CN, SC, SR

CN**COSINE ON**

The CN command enables cosine velocity ramps; i.e. half sinusoid acceleration profiles, for all axes. The cosine is not truncated in moves that do not reach full speed. This command should not be given while an axis is in motion or the results may be unpredictable. This command affects all axes even if issued in the single axis mode. The PF command is used to return to linear motion profiles. See the [AP Command on page 5-10](#) to preserve the CN setting as the Power up/Reset ramp.



Example: Set the board to be in cosine mode.

Enter: CN

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
CN	AX – AT	29	29
CN	AA-AM	29	29
-	AA/CD	Not Valid	

Related commands: ?RT, SC, PF, LA, PN, PR, SR

SC**COSINE RAMP PER AXIS**

The SC command specifies that the standard cosine acceleration ramp is to be used by the selected axis/axes. This command is similar to the CN command except that only the selected axis or axes are affected. The AP command will store the settings of SC as the power-up/reset defaults.



Example: Select the cosine ramp for the X axis.

Enter: AX SC



Example: Select the cosine ramp for the Y and T axes.

Enter: AA SC,1,,1;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
SC	AX – AT	29	29
SCb,b,b,b;	AA-AM	29	29
-	AA/CD	Not Valid	

Related commands: ?RT, CN, PF, LA, PN, PR, SR

SR SELECT CUSTOM RAMP



The SR command selects a previously defined custom ramp profile for use with a specific axis. This command will override previous ramp type selection for the given axis such as PN and CN.

RANGE: 0 <= SR <= 9



Example: Select custom ramp number 4 for use with axis Y and ramp number 8 for axis T.

Enter: SR, 4, , 8;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
SR#;	AX – AT	12	60*
SR#,#,#,;	AA-AM	12	60
-	AA/CD	Not Valid	

* (10+(2 for each segment in the ramp))

Related commands: ?RT, PF, LA, PN, PR, CN, SC, ED, DAR, DAB, DAE

?RT**REPORT RAMP TYPE**

The ?RT command reports the current acceleration ramp assigned to the active axis. Possible responses are:

LA Default linear ramp
 PRn Parabolic where n specifies number of segments
 SC Cosine ramp
 SRn Custom ramp where n specifies the table number



Example: Make sure custom ramp #3 was assigned to the Y axis

Enter: AY ?RT

Response: <LF><CR>SR3<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?RT	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: PF, LA, PN, PR, CN, SC, SR

5.8.4. CUSTOM PROFILE COMMANDS

When linear, parabolic, and cosine acceleration ramps are insufficient, custom ramps can be defined to meet virtually any profiling need. The following commands provide the capability to define almost any conceivable profile.

DAR

BEGIN CUSTOM RAMP DEFINITION



The DAR command starts the definition of a custom ramp table. A parameter supplied with this command, from 1 to 8, specifies which ramp table to create. If a ramp table by that number has already been defined, it will be overwritten.

Once the DAR command has been issued, only the DAB and DAE commands will be valid. A series of ramp table breakpoints may be entered using the DAB command which define the profile breakpoints for this ramp table. Up to 25 breakpoints may be defined but a smaller number may be used. A ramp table containing no breakpoints is invalid and will result unpredictably if used.

RANGE: 1 <= DAR <= 8



Example: Create a ramp table definition resembling a jerk-limited linear profile.

Enter:

```
DAR3;          * Store as table #3
DAB.1, .05;    * Ramp at 10% of AC until 5% of VL
DAB.3, .1;     * Ramp at 30% of AC until 10% of VL
DAB.9, .9;     * Ramp at 90% of AC until 90% of VL
DAB.3, .95;    * Ramp at 30% of AC until 95% of VL
DAB.1, 1;      * Ramp at 10% of AC until 100% of VL
DAE           * End table definition
```

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
DAR#;	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: DAB, DAE, ED, ?DA, ?DS, ?DE, SR

DAB DEFINE CUSTOM RAMP BREAKPOINT



The DAB command sets a breakpoint in a custom ramp table. This is the only command that should be used after DAR and before DAE. Each custom ramp may contain up to 25 breakpoints, each defined by a DAB command.

The DAB command takes two parameters; the first specifies the acceleration level that should be used to achieve the second parameter, velocity level. Both levels are expressed in terms of percentage in decimal format; i.e. 1.00 is 100%. At no time should a DAB command be entered in which the velocity parameter is less than the velocity parameter of the prior DAB. The PC1x will not flag this as a command error but the results of such a ramp will be unpredictable. Each DAB command sent should be equal to or greater than the DAB command that preceded it. It is the user's responsibility to make sure this command is used properly.

RANGE:

0.00000 <= Parameter 1 <= 1.00000

0.00000 <= Parameter 2 <= 1.00000



Example: See the [DAR command on page 5-136](#) for a complete example of a custom profile.

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
DAB#,#;	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: DAR, DAE, ED, ?DA, ?DS, ?DE, SR

DAE **END CUSTOM RAMP DEFINITION**

The DAE command terminates a custom ramp table definition initiated by the DAR command.



Example: See the [DAR command on page 5-136](#) for a complete custom ramp table definition.

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
DAE	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: DAR, DAB, ED, ?DA, ?DS, ?DE, SR

?DA**PRINT A CUSTOM RAMP**

This command will print out a previously defined custom ramp table. If an undefined table number is specified, only linefeed + carriage return pairs will be returned.

RANGE: 1 <= ?DA <= 8



Example: Print out custom ramp table #2

Enter: ?DA2 ;

Response: <LF><CR>DAR2<LF><CR>
 DAB0.10000,0.20000<LF><CR>
 DAB0.90000,0.80000<LF><CR>
 DAB0.10000,1.00000<LF><CR>
 DAE<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?DA#;	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: DAR, DAB, DAE, ?DS, ?DE

?DS**REPORT THE SIZE
OF A CUSTOM RAMP TABLE**

The ?DS command returns the size of the specified custom ramp table as the number of breakpoint entries in the table.

RANGE: 1 <= ?DS <= 8



Example: The 3rd custom ramp should be 17 breakpoints long. Make sure this is true.

Enter: ?DS3 ;

Response: <LF><CR>17<LF><CR>

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?DS#;	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: DAR, DAB, DAE, ?DA, ?DE

?DE**REPORT A CUSTOM RAMP
TABLE ENTRY**

The ?DE command will return a specific entry from a specific custom ramp table. The first parameter specifies the table to examine and the second parameter specifies the entry to return from the table.

RANGE:

1 <= Parameter1 <= 8

1 <= Parameter2 <= 25



Example: We can't remember what the 23rd breakpoint in table 4 was set to. Use the ?DE command to find out.

Enter: ?DE4 , 23 ;

Response: <LF><CR> <LF><CR> (there is no 23rd entry in table 4)

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
?DE#,#;	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: DAR, DAB, DAE, ?DA, ?DS

5.9. MOTION GENERATION COMMANDS

5.9.1. JOGGING COMMANDS

When an application requires a motor to move without stopping or, perhaps, to move until told to stop, the jogging commands that follow will be useful. These commands will start motion on an axis, ramping up to the specified jog velocity, and continue indefinitely, stopping only when told to stop, a limit is reached, or a timeout occurs ([see below](#)).

JG
JOG


The JG command is a velocity mode and will move one or more axes at the velocities supplied as parameters. The JG command will accelerate to the programmed velocity at the current AC rate and run until altered by an ST, SA, KL, HF (servo models), another JG command, or a limit switch is encountered while limits are enabled.

The jog velocity may be changed by following the command with another JG command of a different velocity. A change in direction between two JG commands will not cause an axis to ramp to a stop then back up in the opposite direction; rather the axis will switch directions abruptly and ramp up or down to the new velocity. An axis should be stopped with an ST command before reversing directions.

This command modifies the move velocity parameter (VL) for the affected axis or axes. The JG command does not require any other command to start the motion.

RANGE: -1,044,000 <= JG <= 1,044,000



Example: Jog the motor at 100,000 counts per second then change to 35,000 counts per second when the second JG is entered, stay at that velocity for 5 seconds, then stop by decelerating to a stop. Next, jog the motor at 5,000 counts per second in the negative direction.

Enter: JG100000; JG35000; WT5000; ST; JG-5000;



Note: Output events waiting for completion of JG will begin when JG is up to its requested velocity. In this case, the motor will ramp from zero to 100,000, ramp back down to 35,000, flatten out at 35,000 for 5 seconds, then ramp to a stop, before moving in the negative direction at a velocity of 5,000.

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
JG#;	AX – AT	4*	30*
JG#,#,#,#;	AA-AM	4*	30*
-	AA/CD	Not Valid	

* AX-AT/AA-AM

- If PA mode is active add 1
- If the controller has an encoder or servo axis add 1
- PN0 & PR0 ramp selections add 2
- All other PN & PR ramp selections add 9
- CN & SC ramp selections add 9
- SR (Custom) ramp selections add (number of steps in ramp - 1)

Related commands: JF, TM, ST, SA

JF**JOG FRACTIONAL VELOCITIES**

The JF command will jog one or more axes at the velocities specified, like the JG command. The parameter may include a fractional part allowing better resolution at low speeds. The velocity set by this command will remain the default velocity until altered by a VL, JG or another JF command.



Note that, like the JG command, direction of motion should not be changed without first bringing the axis to a stop. Issuing a JF command with a positive velocity followed by another with a negative velocity will cause an abrupt change of direction in the motor.

RANGE: -1,044,000.000 <= JF <= 1,044,000.000



Example: Jog the Y axis at $2 \frac{2}{3}$ steps per second.

Enter: `AY JF2.667;`

or

`AM JF,2.667;`

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
JF#;	AX – AT	4*	6*
JF#,#,#,#;	AA-AM	4*	6*
-	AA/CD	Not Valid	

* AX-AT

- If PA mode is active add 1
- If the controller has an encoder or servo axis add 1

* AA-AM

- If PA mode is active add 1
- If the controller has an encoder or servo axis add 1

Related commands: JG, TM, ST, SA

TM**TIMED JOG**

The TM command performs a jog at the current velocity limits defined for the axis/axes for the specified number of milliseconds. In multiaxis mode, all axes begin moving at the same time and ramp to a stop when their respective jog times have elapsed. The overall jog time will be the parameter passed to the TM command plus deceleration time and acceleration time.

RANGE: 0 <= TM <= 200,000



Example: Jog the X axis for 1000 milliseconds.

Enter: AX TM1000;



Example: Jog the X axis for 1000 milliseconds and the Z axis for 2000 milliseconds, starting both at the same time.

Enter: AA TM1000,,2000;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
TM#;	AX – AT	6*	8*
TM#,#,#,##;	AA-AM	6*	8*
-	AA/CD	Not Valid	

* AX-AT/AA - AM

- If PA mode is active add 1
- If the controller has an encoder or servo axis add 1

Related commands: JG, JF, ST, SA

MO MOVE ONE PULSE

The MO command will output one count in the current direction (do not use the GO command). The direction may be reversed directly by use of the MM or MP command or indirectly via a move such as JG or MR. This command generates the output signal in one sample interval and thus eliminates the latency of generating a ramp with an MR1 GO command sequence.



Example: Move the Z axis one pulse in the negative direction.

Enter: AZ MM MO

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
MO	AX – AT	1	1
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: MM, MP

5.9.2. MOVE SPECIFICATION COMMANDS

The following commands define motions on one or more axes that terminate at specified positions. Full profiles are generated that guarantee position achievement either on a per axis basis or in coordinated fashion.

MA

MOVE ABSOLUTE



The MA command will set up one or more axes to move to the absolute positions supplied as parameters. In AA mode, an axis may remain stationary by entering a comma but omitting the parameter. The move is actually initiated by a GO or GD command (or other G-series commands; see below.)

Each axis will use its predefined acceleration and velocity values to move to the new absolute position. Each axis may or may not get to the destination at the same time because each axis utilizes individual velocities and accelerations. The MT command will ensure all axes reach their target positions simultaneously.



Example: In the single axis mode, move the X axis to absolute position 100,000 counts with the previously entered acceleration and velocity parameters.

Enter: AX MA100000; GO



Example: In the AA mode, move the Y axis to absolute position 10,000 counts and the T axis to absolute position 1,000 counts. The other axes will remain in their current positions.

Enter: AA MA,10000,,1000; GO

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
MA#;	AX – AT	2*	58*
MA#,#,#,;	AA-AM	4*	60*
-	AA/CD	Not Valid	

* AX-AT/AA-AM

- If this axis is using a cosine ramp and the move distance is too short to allow the axis to accelerate to its velocity limit add 11.
- If a previous ML or MT command has made adjustments to the axis velocity and acceleration then:
 - Add 8 plus:
 - PN0 & PR0 ramp selections add 4
 - All other PN & PR ramp selections add 18
 - CN & SC ramp selections add 18
 - SR (Custom) ramp selections add 2 *(number of steps in ramp-1)

Related commands: MR, MT, ML, GO, GD, GS, GU, GN

MR MOVE RELATIVE

The MR command will set up one or more axes to move relative from their current positions at the time the move is executed. In the AA mode, an axis may remain stationary by entering a comma but omitting the parameter. The move is actually initiated by a GO or GD command (or another G-series command; see below.)

Each axis will use its predefined acceleration and velocity values to move to the new position. Each axis may, or may not, get to the destination at the same time, because each axis utilizes individual velocities and accelerations. To ensure all axes reach their destinations simultaneously, use the ML command.



Example: In the single axis mode, move the X axis 2468 steps in the negative direction.

Enter: AX MR-2468; GO



Example: In the AA mode, move the X axis 12345 steps in the positive direction and the Y axis 6789 steps in the positive direction. Both axes will start at the same time.

Enter: AA MR12345,6789; GO

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
MR#;	AX – AT	2*	58*
MR#,#,#,##;	AA-AM	4*	60*
-	AA/CD	Not Valid	

* AX-AT/AA-AM

- If this axis is using a cosine ramp and the move distance is too short to allow the axis to accelerate to its velocity limit add 11.
- If a previous ML or MT command has made adjustments to the axis velocity and acceleration then:
 - Add 8 plus:
 - PN0 & PR0 ramp selections add 4
 - All other PN & PR ramp selections add 18
 - CN & SC ramp selections add 18
 - SR (Custom) ramp selections add 2 *(number of steps in ramp-1)

Related commands: MA, MT, ML, GO, GD, GS, GU, GN

ML MOVE LINEAR

The ML command uses linear interpolation to perform a straight line relative move. Input parameters are relative distances for each axis involved in the move. The ML command should be followed by a GO or GD (or another G-series command; see below) to start the axes together. The velocity and acceleration parameters are scaled to allow the axes to move and finish together. All axes are scaled to the axis with the longest move time. At the end of the move, all involved velocities and accelerations will be restored to their pre-move values.



Example: In the AA mode, move the Y, Z and T axes 10000, 100 and 1000 counts respectively with all axes starting and finishing together. The other axes remain in their previous positions.

Enter: AA ML,10000,100,1000; GO

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
-	AX – AT	Not Valid	
ML#,#,#,#;	AA-AM	4*	60*
-	AA/CD	Not Valid	

* AA-AM:

- If a previous ML or MT command has made adjustments to the axis velocity and acceleration or the velocity and acceleration of this axis must be adjusted to permit the linearly interpolated move:
 - Add 6 plus:
 - PN0 & PR0 ramp selections add 6
 - All other PN & PR ramp selections add 20
 - CN & SC ramp selections add 20
 - SR (Custom) ramp selections add 2 *(number of steps in ramp)

Related commands: MA, MR, MT, GO, GD, GN

MT**MOVE TO**

The MT command uses linear interpolation to move two or more axes to the specified absolute positions. The syntax is similar to the ML command. This command is invalid if loops are being used due to the overhead involved. The command will become valid again after executing an ST or KL command. When used in the contour definition mode, only the axes being used in the contour must be provided for in the MT syntax. A GO or GD command (or another G-series command; see below) initiates the move.

The axis that will reach it's destination first is used as a gauge to modify the acceleration and velocity values of the other axes. This is done to ensure all involved axes arrive at their targets simultaneously. At the end of the move, any velocity or acceleration value that was modified is restored to it's pre-move value.



Example: In the AA mode, move the X, Y and T axes to absolute positions 1000, 10000 and 100 counts respectively, with each starting and finishing together. The unused axis remains in it's previous position.

Enter: AA MT1000,10000,,100; GO

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
-	AX – AT	Not Valid	
MT#,#,##,	AA-AM	4*	60*
MT#,#,##,	AA/CD	6	8

* AA-AM

- If a previous ML or MT command has made adjustments to the axis velocity and acceleration or the velocity and acceleration of this axis must be adjusted to permit the linearly interpolated move:
 - Add 6 plus:
 - PN0 & PR0 ramp selections add 6
 - All other PN & PR ramp selections add 20
 - CN & SC ramp selections add 20
 - SR (Custom) ramp selections add 2 *(number of steps in ramp)

Related commands: MA, MR, ML, GO, GD, GN

5.9.3. MOVE EXECUTION COMMANDS

The following commands initiate moves defined by commands in section 5.9.2 Move Specification Commands. A number of different commands are available, tailored to various application needs.

GO
GO


The GO command will initiate a move which has been previously programmed with such commands as MA, MR, MT, and ML. No operand is required with the GO command. If this command is issued without having defined a move, the results are undefined. Issuing a GO command to execute an already-executed move also has undefined results. Only one GO command should be issued per defined move.



Example: In the single axis mode, move the X axis to absolute position 12345.

Enter: AX MA12345; GO



Example: In the AA mode, move the X axis 2468 steps in the positive direction and the Y axis 2468 steps in the negative direction.

Enter: AA MR2468,-2468; GO

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
GO	AX – AT	5*	13*
GO	AA-AM	7*	15*
-	AA/CD	Not Valid	

* AX-AT/AA-AM

- If the controller has an encoder or servo axis add 3
- If PA mode is active add 2
- If an auxiliary output bit settle time has been specified add 3

Related commands: MA, MR, MT, ML, GD, GS, GU, GN

GD**GO AND RESET DONE**

The GD command may be substituted for a GO command. It will reset the done flags, then initiate the move which has been previously programmed with such commands as MA, MR, MT, and ML just as the GO command does. In single axis mode, only the done flag for the selected axis will be reset.

In AA mode, all the done flags will be reset. In the AM mode, only the axes involved in the move will be reset. This allows the host to reset the interrupts on the axis involved in the next move without affecting other axes which may be still active. Note that this command is probably only useful in applications where commands are queued in advance since the interrupt may be reset before the host has the opportunity to service it if the GD command is waiting in the queue.

If this command is issued without having defined a move, the results are undefined. Issuing a GD command to execute an already-executed move also has undefined results. Only one GD command should be issued per defined move.



Example: In the single axis mode, move the Y axis 12345 counts in the negative direction and set the done flag when the move is completed. Then clear the done flag, move the motor 12345 counts in the positive direction, and set the done flag again when the move is completed.

Enter: AY MR-12345; GO ID MR12345; GD ID



Example: In AA mode, perform a linear absolute move with the X and Y axes to the position 10000,20000 and set the done flag when the move is completed. Then clear the done flag, perform a linear relative move on both axes moving the X axis 10000 steps in the negative direction and the Y axis 20000 steps in the negative direction, and set the flag once again.

Enter: AA MT10000,20000; GO ID ML-10000,-20000; GD ID

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
GD	AX – AT	6*	10*
GD	AA-AM	9*	11*
-	AA/CD	Not Valid	

* AX-AT/AA-AM

- If the controller has an encoder or servo axis add 3
- If PA mode is active add 2
- If an auxiliary output bit settle time has been specified add 3

Related commands: MA, MR, MT, ML, GO, GS, GU, GN

GS**GO AND MONITOR SLIP TRIGGER**

The GS command works exactly like the GO command except that the home switch will be monitored during the motion. If the home switch becomes active the slip flag will be set for the axis. The host application can read the slip flag and see that the home switch was encountered during the move. This is useful in applications that register slip conditions by means other than encoder position verification; in fact, this command is not valid in controls with encoder feedback which includes servo motors.

If this command is issued without having defined a move, the results are undefined. Issuing a GD command to execute an already-executed move also has undefined results. Only one GD command should be issued per defined move.



Example:

Move the X axis 50,000 counts in the positive direction. If the motor slips it will close a switch wired to the home input of the X axis. Monitor this switch during the move and set the slip flag for axis X if the switch becomes active.

Enter:

AX MR50000; GS

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
GS	AX – AT	5*	15*
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

* AX-AT

- If PA mode is active add 2
- If an auxiliary output bit settle time has been specified add 3

Related commands: MA, MR, GO, GD, GU, GN

GU**GO ASYMMETRICAL**

The GU command initiates a previously defined move using the AC value for acceleration and the DC value for deceleration. This command may be used with only one axis at a time; i.e. it is not valid with the ML and MT commands.

If this command is issued without having defined a move, the results are undefined. Issuing a GU command to execute an already-executed move also has undefined results. Only one GU command should be issued per defined move.



Example: Move the Y axis to position 1,500 using the current acceleration and velocity and a deceleration of 5,000 counts per second per second.

Enter: AY DC5000; MA1500; GU

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
GU	AX – AT	3*	11*
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

* AX-AT

- If the controller has an encoder or servo axis add 3
- If PA mode is active add 2
- If an auxiliary output bit settle time has been specified add 3

Related commands: MA, MR, GO, GD, GS, GN

5.9.4. MOVE TERMINATION COMMANDS

The following commands allow termination of move sequences in process. When things go wrong or a motion simply needs to be commanded to stop prematurely, the commands below will be useful. These commands can be used to stop motors gracefully or abruptly, depending on the needs of the application.

ST
STOP


The ST command flushes the queue for the current axis or axes only and causes the axis/axes to decelerate to a stop at the rate previously specified via the AC command. This command is used to stop one or more motors in a controlled manner from jog mode or an unfinished GO or GD command. This command is executed immediately upon receipt. All status and position information is retained. When executed in a multiaxis mode, the ST command is equivalent to the SA command.



Example: Move the Y axis for a while at 1200 steps/second, then ramp to a stop.

Enter: AY JG1200; (wait awhile) ST

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
ST	AX – AT	9*	61*
ST	AA-AM	9*	61*
-	AA/CD	Not Valid	

* AX-AT/AA - AM

- PN0 & PR0 ramp selections add 4
- All other PN & PR ramp selections add 18
- CN & SC ramp selections add 18
- SR (Custom) ramp selections add 2*(number of steps in ramp-1)
- If PA mode is active add 1
- If an auxiliary output bit settle time has been specified add 3

Related commands: SA, SD, SI, SO, KL, KS

SA**STOP ALL**

The SA command flushes all queues and causes all axes to decelerate to a stop at the rate previously specified in an AC command. All status and position information is retained. Even when executed in a single axis mode, this command will cause all axes to stop.



Example: Send all axes on a move, then ramp them to a stop, before they finish.

Enter: AA VL100,100,100,100;
MR1000,2000,3000,4000; GO
(wait awhile)
SA

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
SA	AX – AT	9*	61*
SA	AA-AM	9*	61*
-	AA/CD	Not Valid	

* AX-AT/AA - AM

- PN0 & PR0 ramp selections add 4
- All other PN & PR ramp selections add 18
- CN & SC ramp selections add 18
- SR (Custom) ramp selections add 2*(number of steps in ramp -1)
- If PA mode is active add 1
- If an auxiliary output bit settle time has been specified add 3

Related commands: ST, SD, SI, SO, KL, KS

SD**STOP AND RESET DONE**

The SD command may be substituted for the SA command. It will reset the done flags for all axes, stop all axes at the rates previously specified via the AC command, then flush all axis command queues. This allows the host to be interrupted when all axes have stopped by using the ID command after the SD. The SA ID combination may flag the completion early if one of the axes is already done from a previously executed ID.



Example: Stop all axes and reset all done flags. When all axes have stopped set all done flags.

Enter: AA SD ID

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
SD	AX – AT	10*	58*
SD	AA-AM	10*	58*
-	AA/CD	Not Valid	

* AX-AT/AA - AM

- PN0 & PR0 ramp selections add 4
- All other PN & PR ramp selections add 18
- CN & SC ramp selections add 18
- SR (Custom) ramp selections add 2*(number of steps in ramp - 1)
- If PA mode is active add 1
- If an auxiliary output bit settle time has been specified add 3

Related commands: ST, SA, SI, SO, KL, KS

SI STOP INDIVIDUAL

This command can be used to stop only certain axes. In a single axis mode, the SI command behaves identically to ST. In a multi-axis mode, however, SI can be used to stop any number of axes and can be used in place of SA. Like SA, SI will ramp those axes to be stopped using the rate previously specified via the AC command. This command is useful for stopping a specific axis when the current axis mode is unknown and for stopping several axes without affecting current motion on other axes.

Each parameter represents an axis from X through T. Any non-zero value in a parameter will cause the corresponding axis to be stopped.



Example: Start a motion on all four axes. When input bit 1 becomes true, stop axes Y and T without affecting X and Z.

Enter: AM MR15000,30000,20000,40000; GO
SW1; SI,1,,1;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
SI	AX – AT	9*	61*
SIb,b,b,b;	AA-AM	9*	61*
-	AA/CD	Not Valid	

* AX-AT/AA-AM

- PN0 & PR0 ramp selections add 4
- All other PN & PR ramp selections add 18
- CN & SC ramp selections add 18
- SR (Custom) ramp selections add 2*(number of steps in ramp-1)
- If PA mode is active add 1
- If an auxiliary output bit settle time has been specified add 3

Related commands: ST, SA, SD, SO, KL, KS

SO STOP AT POSITION BY RAMPING FROM DISTANCE



The SO command instructs the PCIX to continue moving until reaching a specified distance (parameter 2) from a specified stop point (parameter 1). The axis will then ramp to a stop within the specified distance. This allows the user to control the point at which deceleration begins, the rate of deceleration, and the stop point, all with a single command.

RANGE:

-33,554,541 <= Parameter 1 (Stop Position) <= 33,554,541
-33,554,541 <= Parameter 2 (Distance from Stop Position to Start Decelerating) <= 33,554,541



Example: The X axis is jogging at 10,000 steps per second. We want the axis to stop at position 50,000 but it must not start ramping until reaching position 46,000.

Enter: SO50000,4000;

Related commands: ST, SA, SD, SI, KL, KS

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
SO#,#;	AX – AT	5*	10*
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

*AX-AT:

- If the controller has an encoder or servo axis, add 1
- If PA mode is active, add 1
- If an AUX bit settling time has been specified, add 3

KL**KILL**

The KL command will flush the command queues and terminate pulse generation of all axes immediately. It is intended for emergency termination of any program and to reset the input queues to a known state.

Step motors may not stop immediately even though no more step pulses are delivered due to inertia of the motor and system load. This may result in slippage of the motor. Therefore, the position counter may not accurately reflect the true position of the motor following this command. All axes should be rehommed to return the position counters to a known state.

Servo motors will be forced into open-loop when this command is issued. This will cause motors to “coast”, possibly resulting in damage to the machine or it's payload. To avoid this and other similar problems in vertical axes, power automatic (PA) mode should be used with axis brakes. Due to the encoders used in servo systems, position will not be lost so rehomming servo axes is unnecessary.

A Control-D (ASCII 4) sent to the PC1x will perform the same function as the KL command. It bypasses the command interpreter and may work when the character buffer is full and the KL command cannot get through the interpreter. A Control-D should be used instead of KL, when the board appears hung-up. This can occur when its input queue is inadvertently filled, by entering a loop sequence that was so long you could not enter the LE command.



Example: Stop all previously defined movement and flush the queue of a partially entered incorrect move command (you wanted a negative move not a positive one), before GO is entered.

Enter: AX MR5000 (oops!) ;KL MR-5000; GO

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
KL	AX – AT	1*	5*
KL	AA-AM	1*	5*
-	AA/CD	Not Valid	

* AX-AT

- If PA mode is active add 1
- If an auxiliary output bit settle time has been specified add 3

* AA-AM

- If PA mode is active add 1
- If an auxiliary output bit settle time has been specified add 3

Related commands: KS, RS, ST, SA, SD, SI, SO, FL

KS KILL SELECTED AXES

This command performs the same operation as the KL (kill) command except that individual axes can be killed without affecting others. KS will flush only the selected axes' command queues rather than the entire board. Refer to the KL command for more details.



Example: The Y axis has hit a limit switch and is now executing commands that were waiting in the queue. This axis must be reset but the other axes must be allowed to continue operation.

Enter: AY KS
 or
 AA KS, 1;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
KS	AX – AT	1*	5*
KSb,b,b,b;	AA-AM	1*	5*
-	AA/CD	Not Valid	

* AX-AT

- If PA mode is active add 1
- If an auxiliary output bit settle time has been specified add 3

* AA-AM

- If PA mode is active add 1
- If an auxiliary output bit settle time has been specified add 3

Related commands: KL, RS, ST, SA, SD, SI, SO, FL

5.9.5. MOVE COMPLETION NOTIFICATION COMMANDS

These commands allow the synchronization of moves with external events or multiple axis sequences. If an application needs to know when a move or series of commands has completed and fully processed, the following commands can be used to generate notifications. Commands are available to generate a simple notification or perform a somewhat more complex analysis to decide when and how to notify the host.

ID INTERRUPT WHEN DONE



The ID command will set the done flag for the axis or axes to which the command was issued. An interrupt to the host will be generated if interrupts have been enabled in the PCIx. In Windows environments, this interrupt is captured by the device driver which will store the flags to pass on to the host application when requested.

This command allows the PCIx to signal the host when a string of commands has been completed. In AA mode, the done flag register bits will be set as each axis encounters the ID in its command stream, but the done flag in the status register will not be set until all axes have executed the ID command. In the AM mode, only the axes active in the most recent move will set their done flags.

Though move commands are most commonly used with the ID command, others may be used as well. The ID command may be sent to the PCIx to tell it to set done flags whenever the host application could benefit from knowing an event or series of commands has completed.



Example: Interrupt the host CPU after the execution of Move Absolute is finished. When the move is finished the ID command will be encountered in the command queue and will set the done flags.

Enter: AX MA100000; GO ID



Example: Wait until an input line becomes false then notify the host that this occurred.

Enter: SW2; ID

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
ID	AX – AT	1	1
ID	AA-AM	1	1
ID	AA/CD	1	1

Related commands: IP, II, IN

II INTERRUPT INDEPENDENT

Like the ID command, the II command tells the PCIX to interrupt the host when each axis finishes a move. Unlike the ID command, only those axes which have been supplied parameters in the most recent move command will get their done flags set and cause interrupts.



Example: The following command sequence would cause interrupts when the Y and T axes finish. If they do not complete at the same time, two separate interrupts would be generated.

Enter: AM MR,1000,,10000; GO II

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
-	AX – AT	Not Valid	
II	AA-AM	1	1
-	AA/CD	Not Valid	

Related commands: ID, IP, IN

IN**INTERRUPT NEARLY DONE**

The IN command allows the PClx to interrupt the host when the axis or combination of axes is nearly complete. When used in an application involving probing a part after a move, the probes could start accelerating down while the stage is finishing its move, improving the overall system throughput. This command is valid in all modes. The IN command must be entered before the GO or GD command since it is executed before the move is complete. The test is only performed during deceleration. If the IN parameter is greater than the ramp down distance, the interrupt will be generated when the control starts decelerating.

RANGE:

-67,000,000 <= Parameter (Distance Before End of Move) <= 67,000,000



Example: The following sequence would interrupt the host when the X axis is complete and the Z axis is within 10,000 counts of being complete. The Y axis completion would be ignored in this example.

Enter:

```
AA
IN0,,10000;
MR100000,100000;GO
MR,,50000; GO
```

Related commands: ID, IP, II

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
IN#;	AX – AT	3	3
IN#,#,#,;	AA-AM	3	3
-	AA/CD	Not Valid	

IP INTERRUPT WHEN IN POSITION

The IP command operates like the ID command except the interrupt is deferred until the stage is within the specified step encoder deadband. If the position hold (HN) is not enabled for an axis, the command will behave like an ID command for that axis.



Example: Set the done flag when axis is within it's deadband.

Enter: AX HV1000; HG100; HD10; HN
 MR1000; GO IP

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
IP	AX – AT	1	1
IP	AA-AM	1	1
-	AA/CD	Not Valid	

Related commands: ID, II, IN

IC INTERRUPT CLEAR

The IC command or the ASCII character Control-Y (hex 19) is used to clear the done and error flags in the status register and the done flag register, otherwise the axis would always appear to be “done”. This command will be executed immediately and will usually be placed in the done and error handler interrupt service routine to clear the interrupt and the associated flags. The Control-Y version of this command is preferred to minimize the latency in its execution. The flags may be polled by an RA or RI command which will also reset the flags.



Note that this command is not necessary in Windows environments using OMS-supplied device drivers and DLLs.



Example: Clear the flags after an X axis move relative of 5000 steps was flagged as done when an ID executes.

Enter: AX MR5000; GO ID (done flag set) IC (done flag cleared)

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
IC	AX – AT	Immediate	
IC	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: CA, GD, ID, IP, II, IN

CA CLEAR AXIS DONE FLAG

The CA command operates like the IC command, except it clears the done flag of the addressed axis only. In multiaxis modes, the CA command clears the flags of all selected axes. Unlike the IC command, CA will not clear other error flags in the status register such as slip and limit.



Example: After a multi-axis move, clear the Z axis done flag only.

Enter: AA MR1000,2000,3000,4000; GO ID
AZ CA



Example: After a multi-axis move, clear the Y and Z axis done flags only.

Enter: AA MR1000,2000,3000,4000; GO ID
CA,1,1;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
CA	AX – AT	Immediate	
CAb,b,b,b;	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: IC, GD, ID, IP, II, IN

5.9.6. VELOCITY STAIRCASING COMMANDS

The following commands describe the velocity staircase mode. This mode is useful in applications requiring a change in velocity at a prescribed position without stopping. Similar to the jogging commands, velocity staircasing will move an axis at a specified velocity. The difference is that the next staircasing command in the queue will not be processed until a specified position is reached. Staircasing also allows the host to specify a position for the motor to stop, unlike the jogging commands.

MP
MOVE POSITIVE


The MP command sets the direction logic to move in the positive direction. The direction output for the current axis will change (if necessary) to reflect the new direction. All subsequent non-direction-specific motion commands will now move in the positive direction.



Example: (see MV command [page 5-170](#))

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
MP	AX – AT	1	1
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: MM, MO, MV, SP, FP

MM**MOVE MINUS**

The MM command sets the direction logic to move in the negative direction. The direction output for the current axis will change (if necessary) to reflect the new direction. All non-direction-specific move commands will now move in the negative direction.



Example: Set the direction line to move in the negative direction on the Y axis.

Enter: AY MM

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
MM	AX – AT	1	1
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: MP, MO, MV, SP, FP

MV**MOVE VELOCITY**

The MV command causes the current axis to move to a new absolute position (parameter 1) at a new velocity (parameter 2). When the destination is reached control will be passed to the next command which should be another MV command or an SP command. If the command is not received in time the controller will continue to move at the specified velocity. Note that this is a slave mode and it is the responsibility of the user to provide the commands in time. They may be queued ahead of time. If a new MV command is sent after the controller has already passed the destination specified in the command, the controller will continue to move at the old velocity.

The PCIX will ramp up or down as needed at the rate previously set with the AC command and travel at the new velocity until the new position is reached. The controller will not reverse direction if the position has already passed, but will behave as explained above. Thus the direction of the move must be specified before starting the move with the MP or MM commands. All destinations must be in absolute position; no position relative moves are allowed due to the nature of these commands. Cosine and parabolic acceleration will not apply.

RANGE:

-33,554,431 <= Parameter 1 (Absolute Position) <= 33,554,431
1 <= Parameter 2 (Velocity) <= 1,044,000



Example: Generate a velocity staircase with the breakpoints given in absolute position. Default acceleration (AC) of 200,000

Enter: MP
 MV10000,30000;
 MV20000,50000;
 MV30000,10000;
 SP35000;

The move as shown in [Figure 5-1 Velocity Staircase Profile](#)

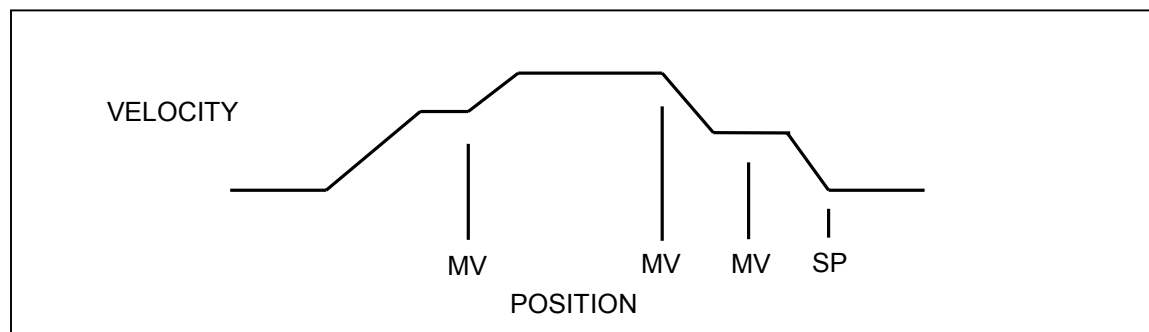


FIGURE 5-1 VELOCITY STAIRCASE PROFILE

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
MV#,#;	AX – AT	4*	6*
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

* AX-AT

- If PA mode is active add 1
- If this controller has an encoder or servo axis add 1

Related commands: MP, MM, SP, FP

SP STOP AT POSITION

The SP command will cause the controller to attempt to stop at the specified destination. If there is insufficient distance to stop at the previously specified deceleration when the command is received, the controller will stop as soon as possible at that deceleration. This command is not compatible with the JG command.

RANGE: -33,554,431 <=SP <= 33,554,431



Example: (see MV command on [page 5-170](#))

Related commands: MP, MM, MV, FP

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
SP#;	AX – AT	3*	8*
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

* AX-AT

- If the controller has an encoder or servo axis add 1
- If PA mode is active add 1
- If an auxiliary output bit settle time has been specified add 3

FP**FORCE POSITION**

The FP command will flush the command queue and attempt to stop at the specified position. The axis will overshoot if there is insufficient distance left to stop at the programmed acceleration. This command should not be given to a servo axis while it is in motion; the results may be unpredictable. Force the axis to stop at a specified position.

RANGE: -33,554,431 <=FP <= 33,554,431



Example: Force axis to stop at 25,000.

Enter: `FP25000;`

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
FP#;	AX – AT	3*	7*
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

* AX-AT:

- If PA mode is active add 1
- If an auxiliary output bit settle time has been specified add 3

Related commands: MP, MM, MV, SP

5.9.7. VELOCITY STREAMING COMMANDS

Velocity streaming is a specialized form of velocity staircasing. Streaming simply produces specified velocities without ramping or other processing. In effect, streaming allows the host to put velocities directly into the PCIX's internal velocity registers for the X and Y axes.

VS
VELOCITY STREAMING


The VS command will generate a pulse train without acceleration or deceleration at the rates specified using the X and Y axes. The parameters are time in 1/1024 second sample intervals, X velocity, and Y velocity. This is a slave mode and cannot be mixed or queued with other commands; i.e. no other motions involving the X and Y axes or AA mode may be currently in execution when this command is issued. AX mode is the only valid mode for use with this command. The VS command does not require a GO command to start the motion; motion begins immediately upon receipt of the complete VS command.

RANGES:

1 <= Parameter 1 (Time in 1/1024 of a second) <= 200,000

0 <= Parameter 2 (X Axis Velocity) <= 1,044,000

0 <= Parameter 3 (Y Axis Velocity) <= 1,044,000



Example:

Create a stair step profile on the X and Y axes, with the X axis moving in the negative direction and the Y axis in the positive direction. Make each step last 1 second long and increase velocity by 1,000 steps/second, until a velocity of 3,000 steps/second is reached, then step down to 0 steps/second.

Enter:

```
AX VS1024,-1000,1000;
VS1024,-2000,2000;
VS1024,-3000,3000;
VS1024,-2000,2000;
VS1024,-1000,1000;
VS1,0,0;
```

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
VS#, #, #,	AX	5	5
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: [See Velocity Staircasing Commands on page 5-168.](#)

5.9.8. CONTOURING COMMANDS

The PCIX will attempt to generate any profile which it is asked to do. It is the responsibility of the host to be sure the acceleration required when generating a circle or any other change in direction is possible within the mechanical constraints of the system. All corners must be defined by arcs and tangents to those arcs, else the change in direction will be instantaneous and generate very large accelerations. The arc radius must be chosen so that the acceleration constraints of the system are met.

Constant velocity contouring is similar to a series of discrete move commands except that it allows multiple discrete moves to be executed in series, without stopping, maintaining a constant vector velocity among the involved axes. The commands below are those that are available in contouring mode.

CV

CONTOUR VELOCITY



The CV command allows specification of contouring velocity. It is executed from the AA mode before a contour definition. A contour defined by a CD command cannot be executed if followed by a CV command. Changing this parameter will make any previously defined contours invalid. The contour velocity defaults to 1000 at powerup or reset. Use WQ between contour definitions to avoid having a CV associated with a second contour definition affect a prior contour still in motion. A CV cannot be issued between a CD and CE command.

RANGE: 1 <= CV <= 1,044,000



Example: (see CD command on [page 5-177](#))

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
-	AX – AT	Not Valid	
CV#;	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: NV, CD

NV**NEW CONTOUR VELOCITY**

The NV command will set a new velocity for a contour currently in execution. When the velocity changes, the PCIX will switch to the new velocity at the start of the next command in the contour queue without ramping. An NV command issued without a contour currently executing will have no effect.

RANGE: $1 \leq NV \leq 1,044,000$



Example: The contour is executing at 25,000 counts per second. Change the velocity to 30,000 counts per second upon execution of the next command in the contour queue.

Enter: NV30000;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
-	AX – AT	Not Valid	
NV#;	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: CV, CD

CD**CONTOUR DEFINE**

The CD command enters contour definition mode and defines a constant velocity contour. The only way to exit this mode is to issue a CE or CK command. Commands following the CD command must be in multiaxis format and be commands valid in CD mode. All multiaxis commands entered in CD mode can address only those axes assigned in the CD command. No commands entered will be executed until a CX command is received which must be issued outside of CD mode.

The CD command takes up to four parameters. Each parameter specifies a starting point for each axis to be involved in the contour in absolute coordinates. If an axis is not to be involved in the contour, its parameter position should be skipped just as it would be in any other multiaxis command. Commands issued within CD mode must be formatted to include only those axes indicated in the CD command. Those that are not in the CD command simply do not exist in the formatting of commands entered in CD mode. For example, if a CD command is issued that uses the X, Y, and T axes (such as CD100,300,,400;), commands entered within CD mode will only consider the X, Y, and T axes. An MT command that moves X to 200, Y to 600, and T to 200 would take the form: MT200,600,200;. Note the lack of a placeholder comma for the Z axis.

Contours that will include circular interpolation (CR) must be defined for only 2 axes in the CD command. Contours involving more than 2 axes may not use the CR command.

When the contour is executed, the PCIX will use the distance between the current position and the contour starting point to linearly ramp up each axis such that all involved axes reach a combined, vectored velocity equal to the value set with the CV command. If this distance is zero, no ramp will be generated resulting in an instantaneous jump to contour velocity. Most stepper systems cannot achieve this and servos will tend to oscillate wildly before settling down if at all. Care should be taken to allow sufficient ramping distance between the contour starting position and the current position when the CX command is issued.

Once the contour is completely executed, the PCIX will ramp the axes to a stop using the rate defined with the AC command. This ramp down will take each axis beyond the final point of the contour. Without manually calculating the ramp down distance for each axis, there is no way to force the contour to come to a complete stop at a predetermined point.

RANGES: -33,554,431 <= CD <= 33,554,431

**Example:**

The following demonstrates cutting a hole with a 10,000 count radius using constant velocity contouring and circular interpolation. The contouring velocity is set to 1000 pulses per second. A contour is then defined beginning at coordinates 0,0 on the X and Y axes. The auxiliary output of the Z axis is turned on, which could turn on the cutting torch or laser starting the cut at the center of the circle. A half circle is cut from the center to the outside of the hole, positioning the cutting tool at the start of the desired hole. The hole is then cut, the torch turned off, the stage stopped and the definition is complete. The stage is then positioned and the hole cut with the CX command. The AN and AF commands must have commas for all axes since they can all be addressed from within the contour definition.

Enter: AA
 CV1000; CD0,0;
 AN,,1; CR0,5000,3.1415926;
 CR0,0,6.2831853;
 AF,1; MT10000,-1000;
 CE
 MT-1000,0; GO CX

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
-	AX – AT	Not Valid	
CD#,#,#,#;	AA-AM	n/a	
-	AA/CD	5 + number of axes in the contour	

Related commands: CV, NV, CR, MT, AN, AF, BL, BH, CE, CK, CX

CR CIRCULAR INTERPOLATION

The CR command defines a move in a circular pattern from the entry position. The first two parameters are the center of the circle in absolute units and the third parameter is the distance to move in radians. Positive radians equal counter clockwise movement. Negative radians equal clockwise movement. The distance parameter should be supplied to seven significant digits if a full circle (2 pi) is to be generated. The radius of the circle is the linear distance between the current position and the first two parameters of the CR command.

The CR command generates a circle by breaking the circle into a series of linear segments. The number of segments will be equal to the total distance traveled divided by the contour velocity divided by 2048 or: distance / (velocity / 2048). If finer resolution is required, the MT command may be used to produce smaller segments but the segments must be calculated by the user.

RANGE:

-33,554,431 <= Parameter 1 (First Coordinate for Center of Circle) <= 33,554,431
-33,554,431 <= Parameter 2 (Second Coordinate for Center of Circle) <= 33,554,431
-6.2831846 <= Parameter 3 (Distance in Radians) <= 6.2831846



Example: (see CD command on [page 5-177](#))

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
-	AX – AT	Not Valid	
-	AA-AM	Not Valid	
CR#,#, #;	AA/CD	8	8

Related commands: CD, MT

MT**MOVE TO**

The MT command causes the axes defined by the CD command to move to the specified absolute position using linear interpolation. Only the axes being used in a contour can be specified in the contouring mode. See the CD command for details.

RANGE: -33,554,541 <= MT <= 33,554,541



Example: Make a hexagon in CV mode using the X and Y axes.

Enter:

```
AA CV5000; CD10000,0;
MT20000,0; MT25000,10000;
MT20000,20000; MT10000,20000;
MT5000,10000; MT10000,0;
CK
CX
```

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
-	AX – AT	Not Valid	
MT#,#,##,	AA-AM	4*	60*
MT#,#,##,	AA/CD	6*	8*

*** AA-AM**

If a previous ML or MT command has made adjustments to the axis velocity and acceleration or the velocity and acceleration of this axis must be adjusted to permit the linearly interpolated move:

- Add 6 plus:
- PN0 & PR0 ramp selections add 6
- All other PN & PR ramp selections add 20
- CN & SC ramp selections add 20
- SR (Custom) ramp selections add 2 *(number of steps in ramp)

Related commands: CD, CR

AF**AUXILIARY OFF**

The AF command may be used within a contour definition allowing control of other devices at any instruction within the contour. The AA mode syntax is used. Any auxiliary can be exercised with this command. All axes must be specified or specifically skipped, rather than those axes defined within the contour, as the other commands in this section. The AF command turns the currently active axis auxiliary port bit off. Note this command will turn power automatic (PA) mode off.



Example: (see CD command on [page 5-177](#))

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
AF	AX – AT	1	1
AF b,b,b,b;	AA-AM	1	1
AF b,b,b,b;	AA/CD	2	2

Related commands: CD, AN, BL, BH

AN AUXILIARY ON

The AN command may be used with a contour by using the AA mode syntax as above. Any auxiliary can be exercised with this command. All axes must be specified or specifically skipped, rather than those axes defined within the contour, as the other commands in this section.



Example: (see CD command [page 5-177](#))

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
AN	AX – AT	1	1
AN b,b,b,b;	AA-AM	1	1
AN b,b,b,b;	AA/CD	2	2

Related commands: CD, AF, BL, BH

CE CONTOUR END

The CE command marks the end of the contour sequence. It will terminate the CD mode and, when executed, ramp to a stop and exit to the AA command mode. The end of the contour should contain at least a short linear segment just prior to the CE command to initialize the parameters for the deceleration of the stage.



Example: (see CD command on [page 5-177](#))

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
-	AX – AT	Not Valid	
-	AA-AM	Not Valid	
CE	AA/CD	2	2

Related commands: CD, CK

CK CONTOUR END AND KILL

The CK command will end the contour sequence, like the CE command, except there is no ramp down; i.e. motion will stop abruptly. It is used in place of the CE command.



NOTE: This command should be used with caution to prevent the stage from slipping or losing its correct position.



Example: Same scenario as CD command, but we want to end the contour with the minimum ramp down.

Enter:

```
AA
CV1000; CD0,0;
AN,,0; CR0,5000,3.1415926;
CR0,0,6.2831853;
AF,0; MT10000,-1000;
CK
MT-1000,0; GO CX
```

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
-	AX – AT	Not Valid	
-	AA-AM	Not Valid	
CE	AA/CD	2	2

Related commands: CD, CE

CX CONTOUR EXECUTE

The CX command will execute the previously defined contour sequence. The stage must be positioned such that it can accelerate to speed by the absolute position specified by the CD command it is executing and must be traveling in the proper direction. Once a contour is defined it may be executed at any time by executing a CX command until it is replaced by another contour definition. The CX command cannot be placed within a loop or while construct.



Example: (see CD command on [page 5-177](#))

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
-	AX – AT	Not Valid	
CX	AA-AM	16*	70*
-	AA/CD	Not Valid	

* AA-AM:

- If the controller has an encoder or servo axis add 1
- If a contour axis has:
 - PN0 & PR0 ramp selections add 4
 - Standard parabolic ramp adds 18
 - Cosine ramp adds 18
 - If PA mode is active add 2
 - If an auxiliary output bit settle time has been specified add 3

Related commands: CD, CE, CK, ID

5.10. SYNCHRONIZATION COMMANDS

5.10.1. WAITING COMMANDS

The commands below provide several methods of command and move synchronization. By forcing the PClx to wait a specified amount of time or wait until a set of axes has stopped moving before processing the next command in the queue, the host gains fine-grained control over the motion process.

WT
WAIT

The WT command will wait for a specified number of milliseconds before proceeding with the next command in the queue. In the AA mode, all axes will wait. Immediate commands will not wait since they are not queued.

RANGE: 1 <= WT <= 32,000



Example: You want to produce pulses on the X axis at 5,000 steps/second for 2 seconds, then 10,000 pulses/second for 3 seconds, then stop.

Enter: AX JG5000; WT2000; JG10000; WT3000; ST;

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
WT#;	AX – AT	3	3
WT#;	AA-AM	3	3
-	AA/CD	Not Valid	

Related commands: WA, WQ, SW, BW

WA **WAIT FOR AXES**

The WA command, valid only in AA mode, allows a command to wait until all moves on all axes are finished before executing any further commands. Some commands which can affect a non-moving axis, such as AN, AF and PA, may execute before a previous move on other axes has finished, especially while in a looping (LS-LE, WH-WG) mode. By preceding these commands with a WA, they will not execute until all previously defined moves have finished.



Example: The Z axis auxiliary line controls a laser beam that you only want on while the Z axis moves in a positive direction. The X and Y axes position the laser. You want to repeat the action 10 times.

Enter: AA VL1000,1000,1000; AC10000,10000,10000;
 LS10; MR1000,1000; GO WA AN,,1; MR,,500; GO AF,,1;
 MR,, -500; GO LE

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
-	AX – AT	Not Valid	
WA	AA-AM	2	2
-	AA/CD	Not Valid	

Related commands: WQ, WT, SW, BW

WQ **WAIT FOR QUEUE TO EMPTY**

The WQ command is a special command that stops the board from processing any new commands until the queue for the current axis mode is empty; i.e. all previous moves have finished. This command is not valid in looping (LS-LE, WH-WG) modes.



Example: Move the Y axis 1,000 steps and wait until the move is complete before asking for the position.

Enter: AY MR1000; GO WQ RP

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
WQ	AX – AT	Immediate	
WQ	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: WA, WT, SW, BW

SW**SYNC WAIT**

The SW command allows synchronization of multi-axis moves or other tasks on one or more PCIX boards by using one of the general purpose input lines. This command causes the PCIX to stop processing new commands until the general purpose input line has been released (allowed to go high) before proceeding with the next command.

The SW command can also be used to cause an axis to wait until the others are finished. To do this, wire-OR the auxiliary lines from several axes together and connect them to a general purpose input line. Use power automatic (PA) mode with the SW command on that line. All commands after that will wait until all axes release their auxiliary lines; i.e. come to a complete stop.



Example: The following command sequence will cause the X axis move to wait until the Y axis has finished its move and turned off its auxiliary output which has been wired to the general purpose input 0 line.

Enter: AY AN MR2000; GO AF
 AX SW0; MR10000; GO

The SW command provides a way to synchronize moves on two or more boards. The following example shows one way to do this.



Example: You have 3 four axis boards, for a total of 12 axes to move together. Call board 1 the “master” and boards 2 and 3 the “slaves”. Wire board 1’s X axis auxiliary line to the two slave boards’ general purpose input 0 line. Send to the master the command “AX PA0”, setting the master’s X axis auxiliary line low until its move starts. This also sets the slaves’ general purpose input 0 line low. Enter the “SW0” command to the two slaves, followed by the move and GO commands. On the master, enter the move command, followed by the GO command. When the master’s move starts, the PA command will set the auxiliary line high releasing the wait on the slave boards. All three boards will start their moves. This provides synchronization to within 480us of each board.

Procedure: Wire board 1’s X axis auxiliary line to board 2’s and board 3’s general purpose input 0 line.

Enter: (Board 1) AX PA0;
 (Board 2) AA SW0; MR200,200,200,200; GO
 (Board 3) AA SW0; MR300,300,300,300; GO
 (Board 1) AA MR100,100,100,100; GO

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
SW#;	AX – AT	2	2
SW#;	AA-AM	2	2
-	AA/CD	Not Valid	

Related commands: WA, WQ, WT, BW

BW**WAIT FOR INPUT TO GO LOW**

The BW command is just like the SW command except that it waits for the input line to reach a TTL low rather than a TTL high. Refer to the SW command for more detail.



Example: See the examples for the SW command

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
BW#;	AX – AT	2	2
BW#;	AA-AM	4	4
-	AA/CD	Not Valid	

Related commands: WT, WA, WQ, SW

5.11. LOOPING COMMANDS

5.11.1. LOOPING COMMANDS

Often, applications have need of the ability to repeat a sequence of commands until some event occurs. The commands in this sections will allow looping over a given series of commands until a timeout or an I/O event such as a rising or falling edge occurs or simply to loop a specific number of times. It should be noted that only queued commands can be looped; immediate commands will be executed immediately and will not stay in the queue to be part of a loop.

LS

LOOP START



The LS command sets the loop counter for the axis being programmed in the single axis mode and all axes in the AA mode. The command expects a loop counter operand following the command. The commands up to the LE loop terminator will be executed the number of times specified by the operand. Loops may be nested up to four levels deep on each axis. The parameter must be less than 32,000.

The first loop of commands will occur immediately as they are entered. The remaining loops will be executed after the loop terminator (LE) has been entered.

The axis mode (e.g. AX, AY, AA) must be the same when entering and exiting the loop, otherwise the matching loop termination command will not be found by the board's command processor. The axis mode may be switched within the loop provided the board is in the same mode when the LE command is sent as when the LS command was sent.

If you want one axis to wait for another in the loop, you must be in the AA mode throughout the loop. If you are in the single axis mode in the loop, each axis' commands will go into their separate queues and execute independently of each other.

Another important thing to note is that the command queue size is 800. Each queued command takes one or more slots. If, when entering a looping sequence of commands, all 800 queue slots are filled before the LE loop terminator is entered, the board will hang. This is because there is no space for the LE command or any other commands. To clear this hang up, send the board a Control-D (same as KL, but shorter) to kill all moves and flush all queues. When programming a loop of more than four or five moves, the queue size should be examined with the RQ command to see if it is nearing zero.

RANGE: 1 <= LS <= 32,000



Example: Execute a 100,000 count relative move on the Z axis 5 times.

Enter: AZ LS5; MR100000; GO LE



NOTE: The first move will occur immediately after entering the GO command. The remaining 4 moves will be executed after the loop terminator LE has been entered.



Example: Execute a 100,000 count move relative on the X axis together with a 100 count move on the T axis, followed by a move absolute to 100 counts on the X axis and 200 counts on the T axis, four times.

Enter: AA LS4; MR100000,,,100; GO MA100,,,200; GO LE

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
LS#;	AX – AT	2	2
LS#;	AA-AM	2	2
-	AA/CD	Not Valid	

Related commands: LE, WH, WS

LE LOOP END

The LE command terminates the most recent LS command. The axis will loop back and repeat the commands within the loop the number of times specified in the LS command. The loop will start repeating as soon as this command is issued.



Example: Perform a relative move on axis Z 5 times. After each Z move, wiggle the T axis 20 times.

Enter: AZ LS5; MR5000; GO
 AT LS20; MR50; GO MR-50; GO
 LE (terminates the LS20; command)
 AZ LE (terminates the LS5; command)

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
LE	AX – AT	2	2
LE	AA-AM	2	2
-	AA/CD	Not Valid	

Related commands: LS

WS**WHILE SYNC**

The WS command will execute the commands between the WS and WD commands as a loop while the specified general purpose input line is true; i.e. low. When the line goes high the PClx will exit the loop and execute the commands which follow the WD command. The test is at the bottom of the loop; i.e. the commands between WS and WD will always be executed at least once.

If the input line specified is already high when the WS/WD loop is issued to the PClx, those commands will be executed only once.



Example: Execute a continuous loop, moving the X axis 10,000 counts and then move the Y axis -1000 counts, until an external device terminates the loop by setting general purpose input 1 high.

Enter: AA WS1; MR10000; GO MR,-1000; GO WD

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
WS#;	AX – AT	2	2
WS#;	AA-AM	2	2
-	AA/CD	Not Valid	

Related commands: WD, LS, WH

WD**WHILE END**

The WD command serves as the loop terminator for the WS command.



Example: ([see WS command on page 5-194](#))

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
WD	AX – AT	2	2
WD	AA-AM	2	2
-	AA/CD	Not Valid	

Related commands: WS

WH**WHILE**

The WH command will execute all commands between it and the terminating WG command as a loop until terminated by a CW command. This allows repeated execution of a command sequence which can be terminated by the host. These commands may not be nested but may be executed sequentially.



Example: You have a 3 axis platform that you use to drill holes in the center of a ¼ inch thick sheet of metal. The sheet is 6 inch square. The driver / motor / lead - screw pitch provide 10000 steps per inch. The operator must manually insert and remove the square from the platform. The X and Y axes move a drill into the desired position. The Z axis lifts and lowers the drill. The operator presses a switch which tells the motion controller that the square is in place and ready to be drilled. The operator will continuously remove and replace the squares until ready to take a break.

The following is a description of how to set up an OMS board to perform this task.

Procedure: Connect a normally closed momentary switch between user I/O input line 0 and ground. This will be the "Ready to Drill" switch.

Enter:

```

AX UU10000;      *set up user units so we can
                  reference move to inches
AY UU10000;      *10000 steps = 1 inch
AZ UU10000;
AX VL.1; AC10;    *set up X axis homing velocity and
                  acceleration
AY VL.1; AC10;    *set up Y axis homing velocity and
                  acceleration
AZ VL.1; AC10;    *set up Z axis homing velocity and
                  acceleration
AX HR; AY HR; AZ HR; *send each axis to home
AA VL3,3,.5;      *set normal move velocity for X, Y
                  and Z axes
WH               *start of loop to drill squares
                  indefinitely
                  *(operator removes/replaces square
                  into platform)
SW0;             *wait until operator presses switch
MA3,3; GO        *move to center of square
MA,,.5; GO       *move the drill through the square (a
                  1/2 inch move on the Z axis drills
                  through the square)
MA,,0; GO        *lift the drill
MA0,0; GO        *move the platform to home position
WG              *loop back to starting WH command
(CW)            *operator wants a break so he/she
                  sends CW from keyboard and presses
                  switch once more (since loop will
                  most likely be waiting for the
                  switch at this point)
                  *the loop ends and the following
                  commands execute
MA0,0,0; GO      *move to home position
  
```

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
WH	AX – AT	3	3
WH	AA-AM	3	3
-	AA/CD	Not Valid	

Related commands: WG, CW, LS, WS

WG**WHILE FLAG**

The WG command serves as the terminator for the WH command.



Example: (see WH command [page 5-196](#))

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
WG	AX – AT	2	2
WG	AA-AM	3	3
-	AA/CD	Not Valid	

Related commands: WH, CW

CW**CLEAR WHILE**

The CW command breaks the WH command upon execution of the remaining commands in the loop; i.e. the current execution of the loop is finished. The WH loop is always executed at least one time since the test for the flag is at the bottom.



Example: (see WH command [page 5-196](#))

QUEUE REQUIREMENTS			
FORMAT	MODE	Min	Max
CW	AX – AT	Immediate	
CW	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: WH, WG

This page intentionally left blank

6.

HOST SOFTWARE

6.1. INTRODUCTION TO PCIX SOFTWARE SUPPORT

A disk containing device drivers, application software, and demonstration code for Oregon Micro Systems PCIx family controllers is supplied with the initial purchase of a PCIx controller. Refer to the text files on the disk for installation instructions and other information.

Some programs on the demo disk that include source code may be adapted for use in application programs that use OMS motion controls. No license is required.

This page intentionally left blank

7. SERVICE

7.1. USER SERVICE

The PCIx family of controllers contain no user serviceable parts.

7.2. THEORY OF OPERATION

The 68332 microprocessor on the PCIx controller maintains four concurrent processes. The highest priority process calculates the desired velocity 2048 times each second with a proprietary algorithm (patent number 4,734,847). This frequency is written to logic on board which generates the pulses for stepper motor control and/or the appropriate voltage levels for Servo Control. The velocity profile and synchronization of each axis is also handled by the 68332.

The commands from the PCI computer are temporarily stored in a 124 character buffer until the 68332 microprocessor can parse them. The command is then executed immediately or routed to separate command queues for each axis. The command queue contains a list of addresses to execute followed by an optional parameter. A command from the host may be expanded into several commands to the appropriate axis. The GO command, for example, will expand into start, ramp up, constant velocity and ramp down commands. The LS command will save its parameter, the loop count, on a loop stack along with the address of the LS command to be used by the next LE command as a target for a jump command. The LE command will decrement the loop count and jump to the most recent LS command providing the loop count has not reached zero. If the loop count has reached zero and it is not nested inside another loop, the queue space will be flagged as available and the next instruction in the queue will be executed.

The communication interface is performed by a dedicated IC on the PCIx. It is a specialized IC for PCI communication and controls the passing of the register information and data to and from the PCIx CPU. Interrupts from the PCIx to the PCI host are generated by this component. Status of the interrupts and error flags may be read by the host in the status register.

This page intentionally left blank

APPENDIX A.

LIMITED WARRANTY

The Seller warrants that the articles furnished are free from defect in material and workmanship and perform to applicable, published Oregon Micro Systems, Inc. specifications for one year from date of shipment. This warranty is in lieu of any other warranty express or implied. In no event will Seller be liable for incidental or consequential damages as a result of an alleged breach of the warranty. The liability of Seller hereunder shall be limited to replacing or repairing, at its option, any defective units which are returned f.o.b. Seller's plant. Equipment or parts which have been subject to abuse, misuse, accident, alteration, neglect or unauthorized repair are not covered by warranty. Seller shall have the right of final determination as to the existence and cause of defect. As to items repaired or replaced, the warranty shall continue in effect for the remainder of the warranty period, or for 90 days following date of shipment by Seller of the repaired or replaced part whichever period is longer. No liability is assumed for expendable items such as lamps and fuses. No warranty is made with respect to custom equipment or products produced to Buyer's specifications except as specifically stated in writing by Seller and contained in the contract.

This page intentionally left blank

APPENDIX B.

TECHNICAL SUPPORT

Oregon Micro Systems , Inc. can be reached for technical support by any of the following methods:

- | | |
|---------------------|--|
| 1. Internet E-Mail: | support@omsmotion.com |
| 2. World Wide Web: | www.omsmotion.com |
| 3. Telephone: | 8:00 a.m. - 5:00 p.m. Pacific Standard Time
(503) 629-8081 or (800) 707-8111 |
| 4. Facsimile: | 24 Hours
(503) 629-0688 or (877) 629-0688 |
| 5. USPS: | Oregon Micro Systems Inc
1800 NW 169th Place Suite C100
Beaverton OR 97006 |

RETURN FOR REPAIRS

1. Call Oregon Micro Systems Customer Service at 503-629-8081 or (800) 707-8111 or E-Mail to sales@omsmotion.com.

2. Explain the problem and we may be able to solve it on the phone. If not, we will give you a Return Materials Authorization (RMA) number.

Mark the RMA number on the shipping label, packing slip and other paper work accompanying the return. We cannot accept returns without an RMA number.

3. Please be sure to enclose a packing slip with the RMA number, serial number of the equipment, reason for return, and the name and telephone number of the person we should contact if we have further questions.
4. Pack the equipment in a solid cardboard box secured with packing material.
5. Ship prepaid and insured to:

OREGON MICRO SYSTEMS, INC.
Twin Oaks Business Center
1800 NW 169th Place, Suite C100
Beaverton, OR 97006

This page intentionally left blank

APPENDIX C.

SPECIFICATIONS

Velocity

0 to 1,044,000 counts per second
simultaneous on each axis

Acceleration

0 to 8,000,000 counts per second per
second

Position range

67,000,000 counts ($\pm 33,500,000$)

Accuracy

Position accuracy and repeatability ± 0
counts for point to point moves

Velocity accuracy $\pm 0.01\%$ for step pulse
output

Environmental

Operating temperature range: 0 to 50
degrees centigrade

Storage temperature range: -20 to 85
degrees centigrade

Humidity: 0 to 90% non-condensing

Power

+5VDC at 1 amp typical

+12VDC at 0.1 amp typical

-12VDC at 0.1 amp typical

Dimensions

Conforms to dimensions for 5V PCI Raw
Card (Short Card) in PCI Specification
Rev. 2.2

Communication

Meets all signal specifications for PCI
Specification Rev. 2.2

Limit switch inputs

Opto-isolated TTL input levels with on board
270 Ω current limiting resistor, requires only
external switch closure to ground or TTL
level input signal. Input sense (low or high
true) selectable by command input for each
axis.

Home switch inputs

Opto-isolated TTL input levels with on board
270 Ω current limiting resistor, requires only
external switch closure to ground or TTL
level input signal. Input sense (low or high
true) selectable by command input for each
axis.

User definable I/O

Up to 12 bits of user definable I/O. Factory
default sets 4 bits as outputs and 4 bits as
inputs. One auxiliary output per axis is
included and these are fixed as outputs.
General purpose I/O and auxiliary bits are
optically isolated. Inputs include an on board
270 Ω current limiting resistor. These inputs
expect TTL level input signals. I/O Output
bits and auxiliary outputs are optically
isolated TTL signals with 2.2K Ω pull-up
resistors on board (opto, max 50mA).

Analog outputs

+/-10V and 0 to +10V

Step pulse output

Pulse width 50% duty cycle. Open collector
TTL level signal (7406, max 48mA).

Direction output

Open collector TTL level signal (7406, max
48mA).

OMS PCIX(PCI) INTELLIGENT MOTION CONTROLS				
MODEL	SERVO AXES	STEPPER AXES		USER I/O
		CLOSED LOOP (Encoder)	OPEN LOOP	
PCIX-002			2	10
PCIX-020		2		10
PCIX-202	2		2	12
PCIX-200	2			10
PCIX-004			4	12
PCIX-040		4		12
PCIX-400	4			12

This page intentionally left blank

INDEX

?

?AC	5-125
?AD	5-45
?DA	5-139
?DB	5-76
?DE	5-141
?DS	5-140
?DZ	5-71
?ER	5-92
?ES	5-100
?HD	5-94
?HG	5-98
?HV	5-96
?KA	5-78
?KD	5-86
?KI	5-84
?KO	5-73
?KP	5-82
?KV	5-80
?LS	5-58
?PA	5-38
?PM	5-89
?RT	5-135
?SE	5-42
?SL	5-61
?SO	5-66
?SV	5-69
?TL	5-63
?UU	5-26
?VB	5-123
?VL	5-120

A

AA	5-16
AC	5-124
Acceleration	5-124
Acceleration Feedforward	5-77
ADH	5-43
AF	5-40, 5-181
AM	5-17
AN	5-39, 5-182
AP	5-10
Assign Current Parameters As Power Up Default Values	5-10
AT	5-21
Auxiliary Off	5-40, 5-181
Auxiliary On	5-39, 5-182
AX	5-18

Axes Multitasking	5-17
AXIS T	5-21
AXIS X	5-18
AXIS Y	5-19
AXIS Z	5-20
AY	5-19
AZ	5-20

B

Begin Custom Ramp Definition	5-136
BH	5-47
BI	5-64
Bipolar	5-64
Bit High	5-47
Bit Low	5-46
Bit Request In Hex	5-49
Bit Set	5-48
BL	5-46
BS	5-48
BW	5-190
BX	5-49

C

CA	5-167
CD	5-177
CE	5-183
Circular Interpolation	5-179
CK	5-184
Clear Axis Done Flag	5-167
Clear While	5-199
CN	5-132
Command Table	5-3
Contour Define	5-177
Contour End	5-183
Contour End And Kill	5-184
Contour Execute	5-185
Contour Velocity	5-175
Cosine On	5-132
Cosine Ramp Per Axis	5-133
CR	5-179
CV	5-175
CW	5-199
CX	5-185

D

DAB	5-137
DAE	5-138
DAR	5-136
DBI	5-74
DBN	5-75
DC	5-127

Define Custom Ramp Breakpoint	5-137
Define Zero Position In Open-Loop Mode.....	5-70
Derivative Gain Coefficient.....	5-85
DZ.....	5-70

E

EA.....	5-31
Echo Off	5-15
Echo On	5-14
EF.....	5-15
EN	5-14
Encoder Ratio.....	5-90
Encoder Slip Tolerance.....	5-99
Encoder Status.....	5-31
Encoder Tracking	5-105
End Custom Ramp Definition.....	5-138
ER	5-90
ES.....	5-99
ET.....	5-105

F

FL	5-23
Flush.....	5-23
Force Position	5-173
FP.....	5-173

G

GD.....	5-152
GO.....	5-151
Go And Monitor Slip Trigger.....	5-153
Go And Reset Done	5-152
Go Asymmetrical.....	5-154
GS.....	5-153
GU.....	5-154

H

HD	5-93
HE	5-52
HF.....	5-88
HG.....	5-97
HH.....	5-50
HL.....	5-51
HM.....	5-106
HN	5-87
Hold Deadband	5-93
Hold Gain	5-97
Hold Off	5-88
Hold On	5-87
Hold Velocity.....	5-95

Home.....	5-106
Home And Kill.....	5-109
Home Encoder	5-52
Home High	5-50
Home Low	5-51
Home Reverse	5-108
Home Reverse And Kill	5-110
Home Switch	5-53
HR	5-108
HS	5-53
HV	5-95

I

IC.....	5-166
ID.....	5-162
II	5-163
IN.....	5-164
Integral Gain Coefficient.....	5-83
Interrupt Clear	5-166
Interrupt Done	5-162
Interrupt Independent.....	5-163
Interrupt Nearly Done	5-164
Interrupt On Slip	5-101
Interrupt When In Position.....	5-165
Invert Direction Bit	5-74
Invert Servo Voltage	5-67
IP	5-165
IS	5-101

J

JF	5-144
JG.....	5-142
Jog.....	5-142
Jog Fractional Velocities	5-144

K

KA.....	5-77
KD	5-85
KI	5-83
Kill.....	5-160
Kill Selected Axes.....	5-161
KL	5-160
KM.....	5-109
KO	5-72
KP.....	5-81
KR	5-110
KS.....	5-161
KV.....	5-79

L

LA	5-129
LE	5-193
LF	5-54
LH	5-56
LIMITED WARRANTY	A-1
Limits High	5-56
Limits Low	5-57
Limits Off	5-54
Limits On	5-55
Linear On	5-128
Linear Ramp Per Axis	5-129
LL	5-57
LN	5-55
LO	5-112
Load Motor Position	5-112
LOAD POSITION	5-111
LOOP END	5-193
LOOP START	5-191
LP	5-111
LS	5-191

M

MA	5-147
Macro Execute	5-34
MD	5-32
ML	5-149
MM	5-169
MO	5-146
Move Absolute	5-147
Move Linear	5-149
Move Minus	5-169
Move One Pulse	5-146
Move Positive	5-168
Move Relative	5-148
Move To	5-180
Move To	5-150
Move Velocity	5-170
MP	5-168
MR	5-148
MT	5-150, 5-180
MV	5-170
MX	5-34

N

New Contour Velocity	5-176
Normalize Direction Bit	5-75
Normalize Servo Voltage	5-68
NV	5-176

O

Offset Coefficient In Closed-Loop Mode 5-72

P

PA.....5-36
 Parabolic On.....5-130
 Parabolic Ramp Per Axis 5-131
 PE.....5-118
 PF.....5-128
 PM.....5-35
 PN 5-130
 Power Automatic 5-36
 PP.....5-115
 PR 5-131
 Preserve A Temporary Macro 5-33
 Print A Custom Ramp 5-139
 Print Macro 5-35
 Proportional Gain Coefficient 5-81
 PT 5-33

Q

QA 5-27
 QI.....5-28
 Query Axis 5-27
 Query Interrupt Status 5-28

R

RA 5-29
 RC 5-126
 RD 5-11
 RE 5-117
 Remainder.....5-113
 Report A Custom Ramp Table Entry.....5-141
 Report Acceleration Feed-Forward 5-78
 Report Analog Output Mode.....5-66
 Report Axis Base Velocity Setting 5-123
 Report Axis User Units' Axis Assignment 5-26
 Report Dac Open-Loop Offset 5-71
 Report Default Auxiliary Bit State 5-45
 Report Direction Bit Logic.....5-76
 Report Encoder Positions.....5-118
 Report Encoder Slip Tolerance 5-100
 Report Limit Active State.....5-58
 Report Motor Positions.....5-115
 Report Motor:Encoder Ratio.....5-92
 Report Peak Velocity Setting.....5-120
 Report PID Closed-Loop Offset 5-73
 Report PID Derivative Gain 5-86
 Report PID Integral Gain 5-84

Report PID Proportional Gain.....	5-82
Report PID State	5-89
Report Position In User Units.....	5-116
Report Position Maintenance Deadband.....	5-94
Report Position Maintenance Gain.....	5-98
Report Position Maintenance Velocity.....	5-96
Report Power Automatic State	5-38
Report Servo Voltage Inversion State	5-69
Report Settling Time	5-42
Report Soft Limit Status	5-61
Report Software Travel Limits.....	5-63
Report The Size Of A Custom Ramp Table.....	5-140
Report Velocity Feedforward.....	5-80
Request Acceleration	5-126
Request Axis Status	5-29
Request Encoder Position.....	5-117
Request Interrupt Status	5-30
Request Position	5-114
Request Queue Status.....	5-22
Request Velocity	5-121
Reset.....	5-13
Restore Default Values	5-11
Restore Factory Default Values	5-12
RETURN FOR REPAIRS.....	B-1
Return Slip Status	5-102
RF.....	5-12
RI.....	5-30
RL.....	5-102
RM.....	5-113
RP.....	5-114
RQ.....	5-22
RS.....	5-13
RU.....	5-116
RV.....	5-121

S

SA.....	5-156
SC	5-133
SD	5-157
SE.....	5-41
Set Auxiliary Default To High	5-43
Set Auxiliary To Low.....	5-44
Set Software Travel Limits	5-62
Settling Time	5-41
SF.....	5-60
SI.....	5-158
SL.....	5-59
SO.....	5-159
Soft Limit Off	5-60
Soft Limits On.....	5-59
SP.....	5-172
SPECIFICATIONS	C-1
SR	5-134
ST.....	5-155
Stop.....	5-155

Stop All	5-156
Stop And Reset Done	5-157
Stop At Position	5-172
Stop At Position By Ramping From Distance	5-159
Stop Individual	5-158
SVI	5-67
SVN	5-68
SW	5-189
Sync Wait	5-189

T

TECHNICAL SUPPORT	B-1
Temporary Macro Define	5-32
TF	5-104
Timed Jog	5-145
TL	5-62
TM	5-145
TN	5-103
Turn Off Slip Kill Mode	5-104
Turn On Slip Kill Mode	5-103

U

UF	5-25
UN	5-65
Unipolar	5-65
User Off	5-25
User Units	5-24
UU	5-24

V

VB	5-122
Velocity	5-119
Velocity Base	5-122
Velocity Feedforward	5-79
Velocity Streaming	5-174
VL	5-119
VS	5-174

W

WA	5-187
Wait	5-186
Wait For Axes	5-187
Wait For Input To Go Low	5-190
Wait For Queue To Empty	5-188
WD	5-195
WG	5-198
WH	5-196
While	5-196
While End	5-195

While Flag	5-198
While Sync	5-194
Who Are You	5-9
WQ	5-188
WS	5-194
WT	5-186
WY	5-9